Intelligent Machine Learning Architecture for Detecting DDoS attacks in IoT networks

by

Yahya Sulaiman Al-hadhrami

Thesis submitted in fulfilment of
the requirements for the degree of

Doctor of Philosophy

Under the supervision of **Dr. Farookh Khadeer Hussain**

University of Technology Sydney
Faculty of Engineering and Information Technology

September-2020

Certificate of Authorship/Originality

I, Yahya Al-Hadhrami declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philanthropy (C02029), in the computer science school/faculty of engineering and information technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Production Note:

Signature: Signature removed prior to publication.

Date: 16/09/2020

ABSTRACT

Intelligent Machine Learning Architecture for Detecting DDoS attacks in IoT networks

by

Yahya Sulaiman Al-hadhrami

The Internet of Things (IoT) is growing rapidly across a wide range of applications; one example of such an application is the smart city, in which a city's infrastructure, such as road management, building automation, and people and crowd surveillance, is connected to the Internet. Such applications are being extended to factories, smart agriculture, and even smart devices, which are becoming very common. The rapid growth in the IoT has driven other technologies, such as 5G networks, to grow rapidly to adjust to the sheer number of devices connected to the Internet, and these technologies are expected to further expand the spread of the IoT. However, the existing IoT deployment does not come without challenges, including the large number of connected devices, security issues, and a variety of new standards. From a security perspective, IoT faces a growing threat when it comes to the availability of networks. Distributed denial of service (DDoS) attacks are one well-known threat. However, investigation of the literature shows a lack of solutions with which to tackle DDoS attacks in the IoT.

To address this gap in the literature, this thesis proposes an intelligent machine-learning-based platform that can detect denial-of-service attacks, termed IDD-IoT. The proposed platform consists of several components, including building a real-time dataset generation framework to generate IoT-based datasets (IoT-DDoS) to detect malicious attacks in the IoT, allowing scientists and researchers in the field to further enhance intrusion detection systems with an up-to-date dataset. The

platform then builds on the dataset generation framework, developing an intelligent machine-learning-based framework for detecting three kinds of IoT-DDoS attacks: blackhole, selective forwarding, and flooding attacks. We utilize this framework to build a novel advanced intrusion detection system (IDS) for IoT networks capable of analyzing and detecting DDoS attacks. The IDS consists of a real-time monitoring and analysis unit capable of monitoring traffic in real time with the assistance of an IDS agent that works as a communication link between the IDS and IoT network. We show that our proposed intelligent framework can efficiently detect malicious attacks in respect to security goals such as confidentiality, privacy, and availability, by building an emulated smart IoT environment using the Cooja simulation platform, and we evaluate its performance. Finally, we present the simulation and evolution results to highlight the proposed platform's efficiency, taking into consideration the limitations associated with resource-constrained devices.

thesis directed by Associate Professor Farookh Hussain
School of Computer Science
Faculty of Engineering and Information Technology (FEIT)

Dedication

I dedicate this thesis to my family, friends and all who supported me through this journey.

Acknowledgements

I would like to humbly extend my utmost gratitude and thankfulness to the hands that helped craft my academic endeavours throughout my PhD journey. This journey has been blessed by the Almighty Allah and such blessings were bestowed upon me in the form of people guiding me to the light. One person to whom I must express my sincere appreciation is my superior and principal advisor, Associate Professor Farookh Khadeer Hussain for his endless support. His critical vision and leadership contributed greatly to my work ethic and the outlet it took. You shaped my academic character which will be of great value to me as I begin my career. My journey was also blessed by the encouragement of my family, friends and colleagues to whom I owe my deep gratitude. May Allah give you back the light you brought into my journey. Thank you.

Yahya Al-Hadhrami

Sydney, Australia, 2020.

List of Publications

The following is a list of my research papers produced during my PhD study.

Journal Papers

J-1. **Yahya Al-hadhrami**, and Farookh Khadeer Hussain, "Real time dataset generation framework for intrusion detection systems in IoT," *Future Generation Computer Systems*, 08:414 – 423, 2020, doi:"doi.org/10.1016/j.future.2020.02.051".

J-2. **Yahya Al-hadhrami**, and Farookh Khadeer Hussain, "DDoS attacks in IoT Networks: A ComprehensiveSystematic Literature Review," *World Wide Web Journal*, 2020 (Minor Revision).

Conference Papers

C-1. **Yahya Al-hadhrami**, Nassser Al-hadhrami and Farookh Khadeer Hussain: Data Exportation Framework for IoT Simulation Based Devices. *International Conference on Network-Based Information Systems*, pp.212-222 , Springer, 2019

C-2. **Yahya Al-hadhrami**, and Farookh Khadeer Hussain :A Machine Learning Architecture Towards Detecting Denial of Service Attack in IoT. *Conference on Complex, Intelligent, and Software Intensive Systems*, pp.417-429, Springer, 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Technology is becoming faster and devices smaller each day with a move toward an always-connected model. This revolution means all devices are able to communicate with each other and construct the future internet. This new concept of the future internet is known as the Internet of Things (IoT). IoT is an inter-network of numerous information-sensing objects and services such as infrared sensors, laser scanners, gas indicators, radio frequency identification devices (RFIDs), and global positioning systems (GPS) that can communicate and share information among themselves in different areas of application. Every device, from a cell phone to a car, an alarm clock to a coffee machine, is becoming connected to the internet with advancements in IP addressing schemes. IoT integrates physical things into an information network, and physical devices sense properties such as sound, light, heat, location, etc. and send this data for further processing to a central information network [1].

Today, IoT has several implementation domains, such as environmental, energy, transportation, healthcare, retail, and military. With the rapid IoT adoption, security issues are also emerging due to its heterogeneous nature. Current IoT security issues include privacy protection, heterogeneous network authentication, access control, information storage and management issues, and more. These issues can be broadly classified as communication, distributed denial-of-service (DDoS), node compromise, impersonation, and protocol-specific attacks. DDoS attacks block the service provided by one or more IoT devices behaving maliciously, thereby disabling the network. Therefore, the IoT infrastructure must detect and analyze DDoS attacks properly and swiftly and immediately take the necessary security measures.

The rest of the chapter is organized as follows. Section 1.1 gives a brief overview of IoT and its associated security requirements. In section 1.2, the significance of this

thesis is explained. The structure of the thesis is presented in section 1.3. Finally, this chapter is concluded in section 1.4.

1.1 Background

Note: Some parts of this section have already been submitted to the WWW journal, and it is in its second stage of review [2].

IoT devices and platforms have been growing rapidly in the last few years, and studies have shown that more than 20 billion devices will be connected to the internet by the end of 2020. The unique feature of the growth of these devices is their diversity of applications, ranging from a simple application like a coffee machine connected to the internet to a very complicated mesh of sensors used for industrial purposes. Over the years, IoT has grown to play a significant role in applications such as healthcare and minimizing traffic using the concept of connected vehicles [3].

The accessibility and affordability of IoT devices does not come without consequences, with security being one of the significant consequences of such systems. Due to the lack of a standard architecture for IoT networks and devices, security suffers the most in this heterogeneous network of things [3]. Different vendors have various architecture and protocols; therefore, applying traditional security measures will not deliver the expected result. Moreover, due to the constraints which will be covered in section 1.1, the conventional methods are not applicable due to their high computing needs and resource requirements. In this study, we focus on a security solution based on the systems that have been implemented in the context of IoT devices and networks.

1.1.1 Security Requirements and Goals

Different security protocols are required to achieve a holistic security solution. The commonly used security and assurance model is the CIA triad model, which consists of three requirements:

1. **Confidentiality:** Ensuring sensitive data is protected from unauthorized entities either when the data is in transit or at rest [4]. IoT devices IoT devices

can be used to collect sensitive data, as in the healthcare system, where personal information about the patient is critical and might be life-threatening. In such scenarios, confidentiality is crucial and must not be taken lightly.

2. **Integrity:** Data can be altered when transmitted to the receiver, making the IoT unreliable. Ensuring integrity between IoT devices is essential in most scenarios and applications. The alteration and modification of data while in transit can lead to serious negative consequences in some IoT applications, such as in the health sector where the manipulation of sensitive data, e.g. (blood pressure, heart rate, etc.) could mean life or death for the patients.

3. **Availability:** This is one of the essential security goals as it ensures that the IoT device is accessible at any time when needed [5]. Attacks on availability, usually referred to as denial-of-service attacks (DoS), are a serious threat to any business or organization; as denying access to devices and services can lead to significant losses in business revenue. Therefore, IoT devices and networks must be robust and accessible, even when security threats and attacks are present.

   These three requirements are colloquially referred to as the CIA triad. They have been criticised in the literature as being insufficient to deal with the new threats that emerge every day. To address these limitations, this study proposes a new set of security requirements by analyzing and exploring a broad range of security systems from a security assurance and requirement perspective. The study provides a new list of requirements called the IAS-octave [6].

   This list includes the following four requirements which are additional to the CIA-triad:

4. **Auditability:** This is the process of ensuring that the system is monitoring all its services and actions comprehensively [7]. Auditability might not apply to all IoT applications. Hence it requires more computational resources and storage.

5. **Trustworthiness:** The system's ability to ensure the identity of IoT devices and build trust between the system and third parties [7].

6. **Accountability:** Ensure that each entity in the IoT system is held responsible for each action, which can prevent information misuse [7].

7. **Non-repudiation:** In some cases, the system is required to validate the occurrence of specific actions on a specific event. In the IoT context, such a property might not be considered necessary unless there is some kind of payment involved [7].

In the rest of this study, we use the IAS-octave model for security requirement evaluation.

1.1.2 Constraints and Limitations

Similar other computer networks, IoT networks require various security protocols. However, security measurements for IoT must meet certain criteria that might not apply to different networks due to their nature and resource constraints, which are listed as follows:

1. **Resource Limitation:** One of the challenges facing IoT devices is their limited resources, like CPU and memory, making it very difficult to implement a security solution that requires high processing capabilities [8]. IoT devices are packed with minimal network protocols and minimal features that require less computing power to save energy and resources; therefore, implementing a sophisticated and comprehensive security solution is a constant challenge for manufacturers and developers who need to minimize the number of features and design a simple yet efficient security solution.

2. **Privacy and data Confidentiality:** In IoT, different applications have different privacy implications. The privacy level required for healthcare applications is different from the privacy level required for city temperature sensors. This is not to say that we should neglect privacy concerns for some applications. On the contrary, we should harness IoT devices more where user

privacy is involved. IoT devices have evolved to provide assistance in different aspects of daily life such as smart vehicles and smart homes, which can provide sensitive user information like user location, health status, and user home preferences, all of which raises serious privacy concerns. The key idea of privacy and confidentiality is that the data is kept private and is only accessible by the authorized entity, either a human or machine. To achieve privacy and confidentiality, cryptography is vital and should be applied in a manner that does not affect the IoT constraints and limitations.

3. **Authentication:** IoT devices are generating a huge amount of data every day. The data moving between entities must be securely transmitted and activating the data privacy authentication mechanism is crucial. Unfortunately, there are no common standards used by all vendors for authentication. Different vendors use different authentication protocols which raises security concerns between different platforms since there is no standard authentication. Hence the integration between these platforms is weak and can lead to security issues in the future.

4. **Service Availability:** Service availability in IoT networks is prone to many DOS attacks. Nodes can be compromised internally within the network or from outside intruders; this kind of attack can paralyze the whole IoT network and hinder all activities and services. Moreover, availability attacks usually try to consume all device resources, and since the IoT devices in many cases are battery-powered, this might cause the device to be drained of all its resources. Ensuring that a device or service is available is crucial since many applications are time and data-sensitive, such as those in the healthcare system.

5. **Data Management Challenge:** IoT devices usually generate large amounts of data, and with the increase in the amount of data generated by sensors and devices, data centers face an architectural challenge in coping with such data. Research has shown that the current data centers are not able to handle such an increase in data. IoT at the enterprise level generates a significant amount

of big data that needs to be processed, analyzed, and stored in real-time, which could leave providers with security complications.

1.1.3 IoT Architecture

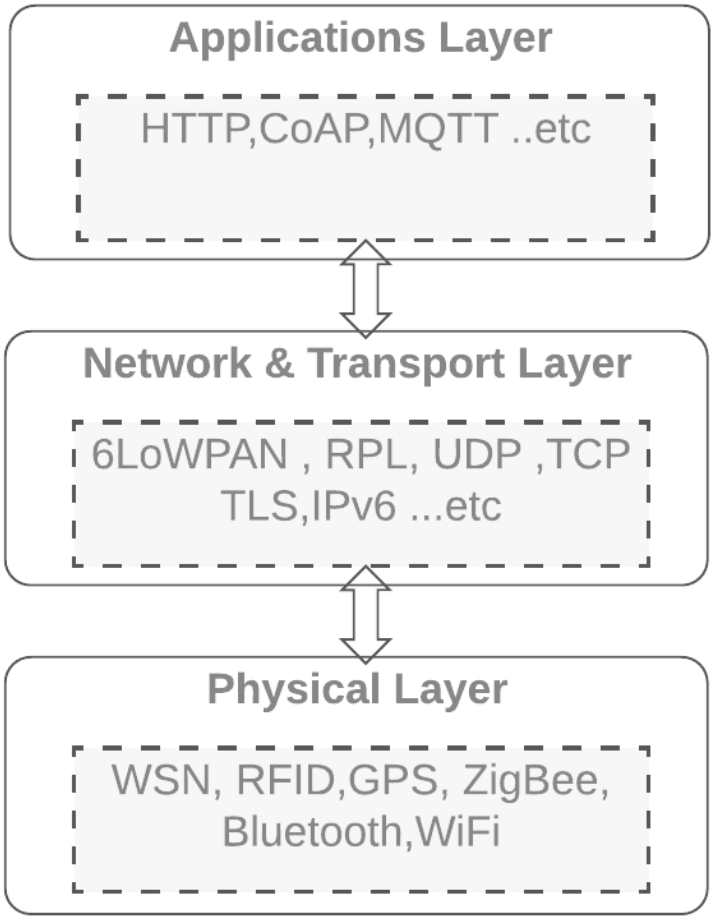


Figure 1.1 : IoT three-layer architecture [9]

Studying and understanding the IoT stack is essential when it comes to building security solutions. Because of the flexibility and different operating systems available for IoT architectures, different system stacks have emerged. This study focuses on Contiki OS, which is widely used and studied by the research community around the world. Figure 1.1 shows the different components of the IoT stack adopted by Contiki OS.

As previously mentioned, different manufacturers have different architecture and unfortunately, there is no standard architecture for IoT devices across different vendors. This thesis focuses on the three-layer architecture since it is the most commonly used by researchers. Figure 1.1 shows the three layers with some examples of the protocols at each level. Each layer is explained briefly as follows:

Physical Layer: This layer responsible for data using sensors and actuators [7]. This layer also responsible for handling node communication, including signal transmission and channel selection, in the case of wireless communication. Some examples of technologies that work in this layer are ZigBee, Bluetooth, and WiFi, 4G/LTE.

Network/Adaptation Layer: This layer is the middle-ware layer that exchanges the data between the application layer and the physical layer. This layer is also responsible for routing the data between different nodes in the network. Moreover, when using the 6lowpan protocol, this layer maps the IPv6 address with the outside world. Some protocols that work in this layer are RPL, 6lowpan, IPv6, and TCP/UDP [3].

Application Layer: This layer is the high-level layer where data representation happens and allows other protocols to access data in the IoT devices like HTTP COaP and MQTT. It is the interaction point between the user and the devices [9].

1.1.4 RPL Protocol

The routing protocol for low power and lossy networks [10] (RPL) routes information between different nodes. It accommodates the many-to-one communication pattern, but it also can support one-to-many and one-to-one communication [11]. The sink node, which is sometimes referred to as the root node, stores the routing information of all nodes. The protocol uses a special directed acyclic graph (DAG) called the DODAG to build the routing tree for communication. Moreover, a special broadcasting message, called the Destination Advertisement Object(DAO), is sent by the root node to allow other nodes to find the network [12]. This message also contains different information about the node, such as the rank. The rank indicates

the current location of the node in relation to its parent and the root node. The rank can be calculated based on different factors that are determined by something called the objective function. The objective function is defined by the network designer to indicate how the route is calculated. It can be calculated by how far the node is from the root or the node's transmission power. The next section discusses the security issues in IoT, specifically the security issues and attacks against the RPL protocol.

1.1.5 IoT Security

Today, many organizations have adopted the IoT paradigm to help them run their businesses more efficiently and to also enhance their personal efficiency. However, IoT faces many security challenges that need to be addressed. Furthermore, different levels of security issues can arise in different layers of the IoT stack, starting from the lower physical layer (802.15.4), where different jamming attacks can be launched to disturb the network, to the upper layers (application), where the attackers can misuse HTTP flood attacks. One of the most pressing and challenging security challenges is the distributed denial-of-service (DDoS) attacks, which can potentially bring down an entire network [13].

DDoS Attacks

A DDoS attack in IoT is the process of compromising IoT nodes and using them to launch a very large-scale attack against other networks or other devices [14]. In this case, the IoT devices work as client nodes for the main attacker device (zombies). Another kind of DDoS attack targets the IoT network itself either from outside (an intruder) or from inside a compromised node with the main goal of paralyzing the IoT network and making it non-functional. For both methods, mitigating DDoS attacks is crucial to maintaining the availability of the network. This research scope covers three kinds of DDoS attacks, which are explained in the following subsections.

Selective Forwarding Attacks

A selective forwarding attack is a denial-of-service attack that targets the IPv6 routing protocol for low-power and lossy networks (RPL) [15]. Its main purpose is to cause a disturbance in the network routing path, where the attacker node selectively forwards only specific packets and drops the others [12]. To launch a selective forwarding attack, the malicious node advertises itself as having a better rank than the parent nodes, causing the adjacent nodes to change their parent and alter the routing path. This kind of attack can be fatal if it is distributed across an IoT network [3].

Blackhole Attack

A blackhole attack is similar to a selective forwarding attack, but instead of forwarding specific packets, it drops all kinds of packets coming from other nodes [16]. This attack also uses the ranking technique to trick the neighboring nodes into dropping all the packets and causing a denial-of-service attack.

Flooding

A UDP flooding attack is extremely popular since it applies to different kinds of networks [17]. The idea of this attack is to use the UDP protocol to frequently send in a forged UDP datagram with a random IP and port, causing the victim to reply to an unknown source, which in this case can cause a denial of service in the victim node. In this thesis, we use a special kind of flooding attack called the DIS Flooding attack, which is explicitly designed for RPL-based networks.

In the next section, we present the existing literature in this space, intending to identify the research gaps.

1.2 SIGNIFICANCE OF THE THESIS

As more devices become connected to the internet every day, the need for a secure and reliable infrastructure increases. IoT encounters many security loopholes that need to be addressed before thinking about the future of IoT. One of these security

issues is DDoS attacks that hinder all of the activity in the network, causing a substantial loss in money and resources [18]. Therefore, addressing this issue is vital to ensure a sustainable IoT future. To the best of our knowledge and through a thorough investigation of the literature, there is no existing solution which uses machine learning as the engine for DDoS detection in IoT.

Therefore, the significance of this thesis is to address this limitation by introducing an intelligent framework for DDoS detection in IoT using machine learning. The further significance of this thesis is listed as follows:

1.2.1 Scientific Significance

1. This is the first research that explores the use of machine-learning approaches to mitigate three types of DDoS attacks (selective forwarding, blackhole, and DIS flooding) in IoT networks.

2. This is the first research that focuses on developing a data collocation tool and building a dataset for the use of this tool in a machine learning detection system in IoT.

3. This is the first research that compares different machine-learning approaches for use in DDoS attack detection in IoT, and to select the optimal approach for DDoS detection in IoT.

1.2.2 Social Significance

1. This study will help IoT consumers use IoT applications more effectively and securely. It also will be a step towards realizing industrial IoT.

2. This study will help service providers to mitigate DDoS attacks more accurately and efficiently. Furthermore, it will help IoT-based service providers to focus on other tasks, which will in time increase productivity.

1.3 Structure of the Thesis

The overall structure of this thesis takes the form of nine chapters. The current chapter introduces IoT and the security challenges associated with it. Furthermore,

the scope of the DDoS attacks and the three attacks involved in this research are briefly presented. The social and scientific significance is also briefly overviewed. The remainder of this thesis is organized as follows:

- **Chapter 2:** In this chapter, a comprehensive literature review following the systematic literature review methodology provides a concise overview of the most relevant studies in the area. In this chapter, different aspects of DDoS attack detection are explored, which are then grouped into four categories based on the type of solution provided. The shortcomings and limitations of each study are then highlighted in a detailed, comprehensive comparison.

- **Chapter 3:** This chapter builds on what has been thoroughly investigated in Chapter 2 and highlights the limitations in the literature and the research gaps. Subsequently, the terms used across this thesis are listed. Finally, the research problem is further divided into four research issues arising from the literature. From these research issues, the research questions and objectives are formulated to reflect the significance of this study.

- **Chapter 4:** This chapter briefly highlights the proposed intelligent system for the DDoS detection framework. Each component of the system is briefly explained, and the system is further categorized into three sections. Each section provides all of the algorithms and technical details of the solution.

- **Chapter 5:** This chapter provides a detailed explanation of the intelligent framework and explores each component individually with an explanation of all of the algorithms. The dataset collection process and the pre-interaction and post-interaction phase are comprehensively presented. This chapter also shows how the data collection model is used to build a new dataset due to the limited datasets in this specific field.

- **Chapter 6:** This chapter explains all the attack implications and network designs, including the integration of the data collection model (DCM). Furthermore, the pre-evaluation tool implementation and detailed analysis of the

RPL protocol is presented. An explanation is given of the real-time data collection, the selected features and the data generation.

- **Chapter 7:** This chapter explains the mathematical model of the three chosen machine learning algorithms and their evaluation in relation to the IoT-DDoS dataset collected in Chapter 6. Then the best performing algorithm is chosen for implementation in Chapter 8.

- **Chapter 8:** This chapter explains the implementation process of the proposed IDS, incorporating the best machine learning algorithms identified in Chapter 7. Also, a comprehensive evaluation for each attack and a detailed comparison with other available solutions are presented.

- **Chapter 9:** This chapter summarizes the thesis and explores the future directions of the research.

1.4 Conclusion

IoT is growing at a rapid pace, involving sectors and domains that were previously not considered. This includes but is not limited to factories, agriculture, cities, and transportation. This wide exploitation of IoT technology and its rapid adaption in a wide variety of sectors poses many security challenges and issues. On of these is the DDoS attack that affects the availability of resources, resulting in an adverse financial impact. This thesis presents an intelligent framework for DDoS attack detection using machine learning, which addresses the issue of the lack of a dataset for machine learning evolution and explores three DDoS attacks in IoT, namely selective forwarding, blackhole and the flooding attack.

Chapter 2

Literature Review

2.1 Introduction

In the previous chapter, IoT and the related security issues were introduced. This chapter presents a comprehensive review of the existing literature on DDoS attack detection in IoT. The three main contributions of this work are summarized as follows: (a) we propose a comprehensive classification of the existing DDoS attacks based on the literature; (b) we detail the systematic approach used to extract all of the existing solutions for DDoS detection in IoT; (c) we report the limitations and weaknesses of the existing methods in the literature. We believe our work provides researchers and knowledge seekers with stepping stones to understand the full picture of the existing security issues in the IoT. The rest of the chapter is organized as follows: Section 2.2 explains the process used to research this chapter. The various types of attacks are defined and categorized in section 2.3. A comprehensive literature review and the limitations associated with the existing literature are presented in section 2.4. Section 2.5 presents a comprehensive discussion and comparison of all of the studies presented. Finally, the chapter is concluded in section 2.6.

Note: The majority of this chapter has already been submitted to the WWW journal and is in its second stage of review [2].

2.2 Research Strategies

This study focuses on building a robust understanding of DDoS attacks in IoT and explores the available solutions to counter such threats. We followed a systematic literature review process to build this comprehensive review.

Numerous review papers have been published in the area of IoT security; therefore, we can form a general idea about the attacks that affect network availability

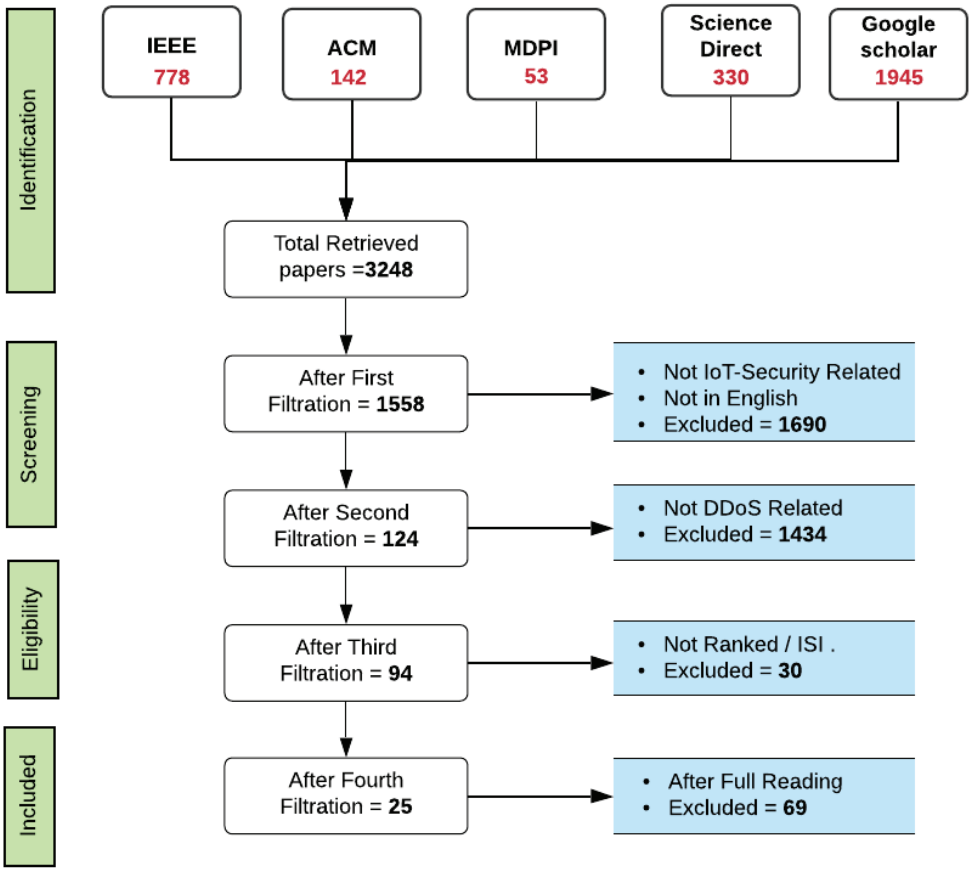


Figure 2.1 : Literature review filtration process

and IoT devices. Table 2.1 shows the review papers that have been thoroughly investigated. Most of the studies covered different aspects of IoT security, yet none specifically address DDoS attacks from an IoT perspective. Therefore, in this systematic literature review, we focus on building a knowledge base of DDoS attacks on IoT and their counter-measure solution.

Table 2.1 : Survey paper comparison

Study	Attacks	New Method	IDS	Protocol	Trust	Authentication	DDoS Specific	Multiple Domain	Security Goals	IoT Architecture	Research Method
[19]	X	X	X	X	X	X	-	-	X	-	-
[20]	-	X	-	X	X	X	-	X	X	-	-
[21]	-	X	-	-	X	-	-	-	X	X	-
[22]	X	-	X	-	-	X	-	-	-	X	-
[23]	-	X	-	-	X	X	-	-	-	-	-
[24]	X	-	X	X	-	X	-	-	X	X	-
[25]	-	-	-	X	X	X	-	X	-	X	-
[26]	X	-	X	-	-	-	-	-	-	X	-
[1]	X	-	X	X	-	X	-	-	X	X	-
[27]	X	-	-	X	X	X	-	X	-	X	-

2.2.1 Keywords

We extracted the following relevant keywords as we only focus on DoS and DDoS attacks. The attacks explored are further explained in section 2.3 of this chapter. To extract the relevant attacks related to our study, the following terms were extracted:

"IoT Security","DDoS attacks IoT","Selective Forwarding IoT" ,"Blackhole Attacks", "Jamming IoT", "6LoWPAN Attack","Flooding Attack".

2.2.2 Research Questions

This study aims to answer the following research questions:

1. What attacks affect the IoT network and cause a denial of services/devices?
2. What are the solutions to counter such threats?
3. How does the proposed solution limit IoT devices, and what security goal does it address?
4. What solutions are mostly used to counter DDoS in IoT?
5. How can the available solutions be categorized?

2.2.3 Research Filtration Process

This section outlines our process for extracting relevant information to obtain data for our review framework. We have gone through the process of filtering the research found in the databases shown in Table 2.2 through the following sequential stages, as shown in Figure 2.1:

- The process starts by collecting the papers based on the keywords defined in Section 2.2.1 from different publishers and databases. Figure 2.1 shows the distribution of the databases.
- The first filtration process is to filter the papers by reading the titles and excluding any paper that is not related to IoT security or is not published in English.
- The second stage of filtration is to filter out any paper that is not related to IoT DDoS attacks. This process was done by reading the title and scanning the abstract if necessary.
- The third filtration stage is to verify that the filtered papers have been ranked or peer-reviewed. We used the CORE database provided by the Computing Research and Education Association of Australasia to check the ranking of the conferences and journal.
- The final stage is to read the papers in full and exclude any that are not explicitly related to the domain of this research, which is DDoS attacks.

Table 2.2 : Databases

Database Name	URL
IEEE	http://ieeexplore.ieee.org/
ACM	http://dl.acm.org/
Science Direct	http://www.sciencedirect.com/
Springer	http://link.springer.com/
Google Scholar	http://scholar.google.com/

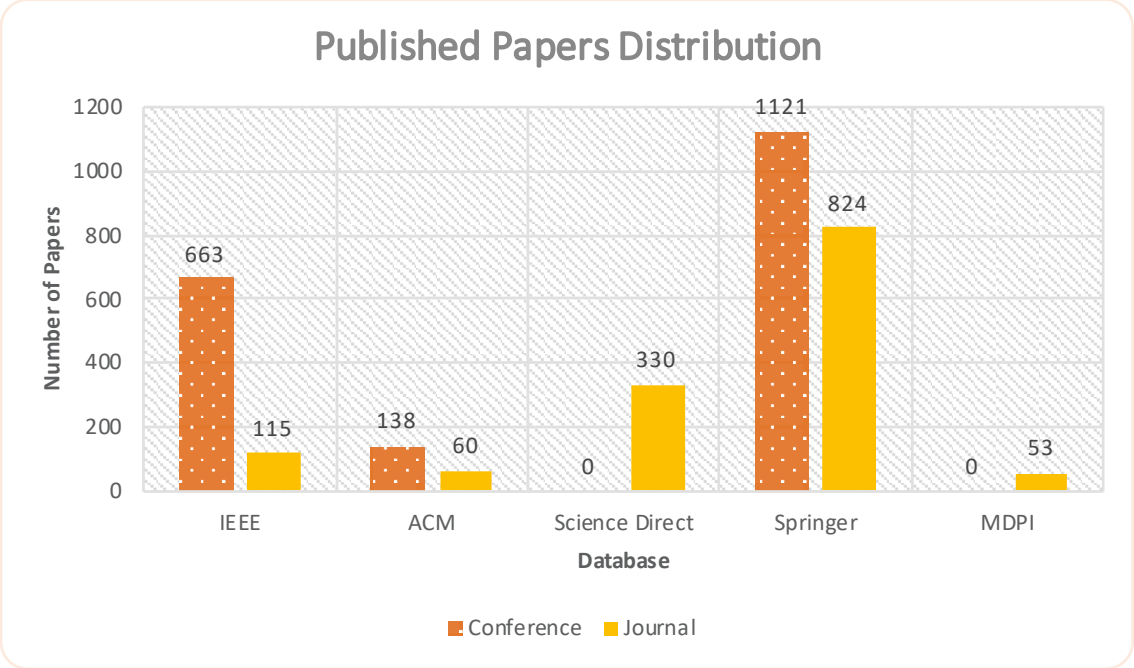


Figure 2.2 : Paper distribution based on database

2.3 Security Issues and Attack Classifications

In reviewing the literature, we identified the attacks responsible for DoS or DDoS attacks, both directly and indirectly. In this study, attack categorization is based on where the attack happens in the IoT three-layer architecture, which is highlighted in this section.

Table 2.3 : Attack summary

Attack	Effected Layer	Effect	Possible Solution	Effected SG
Jamming	PHY	Jamming wireless signal causing DOS	Frequency Hopping	A
Tampering	PHY	Code Modification Maliciously	Physical Security	ACI
Sleep deprivation	PHY	Execution of battery power devices	Split Buffer Solution	A
Unfairness	PHY/MAC	Disturb priority packet sending	-	AI
Collision	PHY/MAC	Exhaustion of battery power devices	Authentication	AI
Buffer Reservation	ADP / 6LOWPAN	DOS using buffer reservation	split Buffer solution	AI
Selective Forwarding	NTW/RPL	DoS and disturbing Network Topology	Authentication, IDS	AIP
Blackhole	NTW/RPL	DoS and disturbing network topology	Authentication, IDS	A
Sinkhole	NTW/RPL	Disturbing network topology, Rank manipulation	Authentication, IDS,	AI
Sybil	NTW/RPL	Masquerading node Identity, compromise privacy	Unique Identifier , Authentication	AIP
Flooding	NTW,PHY,APP	Sending unlimited amount of packets	Authentication, IDS	A
Wormhole	NTW	Routing Distrust	Authentication, IDS	Ai
TCP Hijacking	APP	Stealing node identity using sequence number	Authentication, IDS	AI
6lowpan Fragmentation	ADP / 6lowpan	adding unknown fragment to packet structure	Authentication, IDS	AI

DD - DDoS attack, **Sh**-sinkhole attack, **SF**-Selective Forwarding , **BH**-Blackhole, **HF**-Hello Flood attack, **NTW/RPL** - Network Layer, **L**-Low,**M**-Medium , **H**-High, **D**- Distributed , **C**- Centralized, **U**-Undefined

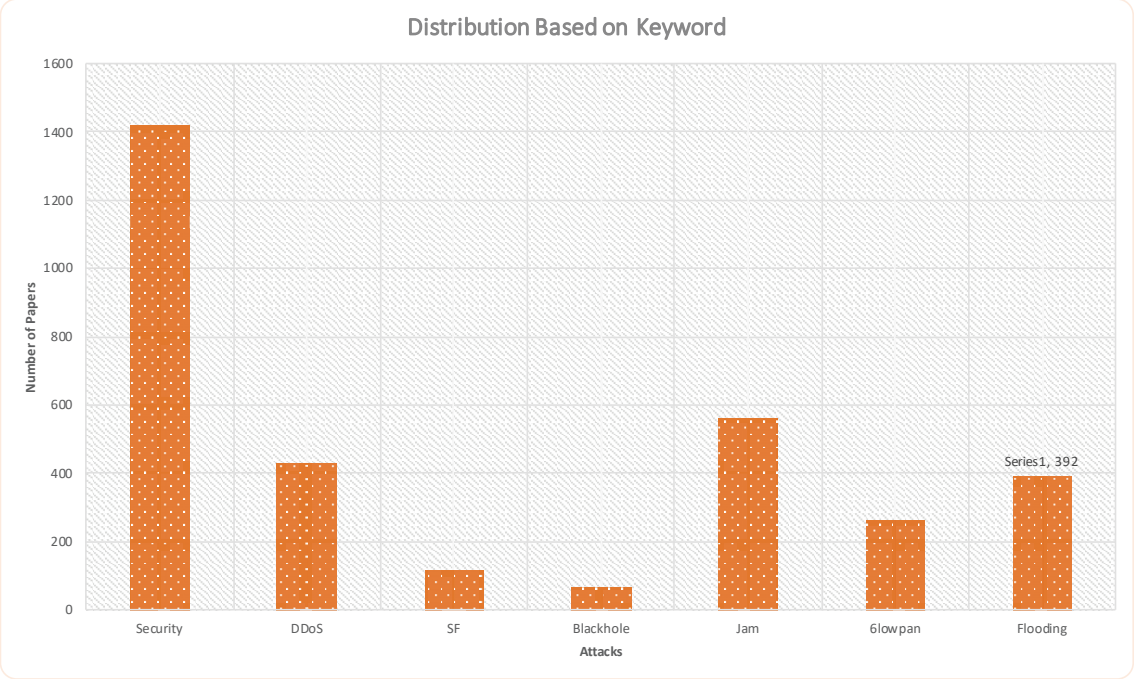


Figure 2.3 : Distribution of papers by keyword

2.3.1 Perception Layer Attack

This layer is a low-level layer where data is acquired. Sometimes it is called the sensing layer since it acquires information from sensing devices such as RFID tags, sensors, or even GPS locations [28]. These devices are usually deployed in unmanned geographic locations where it is easy for intruders to obtain physical access; therefore, these locations are prone to security attacks as outlined in the following.

Jamming

Jamming attacks target the physical layer of the communication stack by interfering with network radio frequencies [8]. This kind of attack is carried out by occupying the same radio frequency channels in the network, which causes node frequency jamming [8]. This attack can be launched against the whole network, causing a large-scale DDoS, hence bringing the entire network down and disrupting all services provided by the network. Moreover, it can target specific network nodes by using a less powerful jamming source [29]. There are many countermeasure so-

lutions against jamming attacks, and a typical defense mechanism is the frequency hopping spread spectrum (FHSS). FHSS is a technique used in signal transmitting by switching between different channels while transmitting. This technique prevents the attacker from knowing which channel is used for node communication.

Tampering

As previously mentioned, IoT devices sometimes are scattered in unmanned areas where devices are unattended. Therefore, such devices are prone to tampering and modifications. An intruder can compromise a node by altering the programming code or injecting malicious code into it. The attacker can go even further and replace the entire node with another node created by the attacker, who can later control it remotely and use it to launch different attacks like blackhole or selective forwarding attacks [29].

Battery Execution (Sleep deprivation)

Battery execution can be achieved by varieties of attacks across the network stack. However, a common attack targets the power saving mechanism in any node, which is called the sleep deprivation attack. It is launched by sending a useless control packet to the victim node, making it forget its sleep cycle until is exhausted and shuts down [30]. This attack is complicated to detect as it typically affects the normal procedure node executions.

Unfairness

In IoT and WSN, there is a feature that allows nodes with low battery life to prioritize packet sending. This feature can be misused by an intruder and impact battery health through the process of forcing nodes with a normal battery level to send priority messages, leading to unfairness in packet sending and disturbing the network behavior [29].

Collision

Data collision occurs when two nodes transmit data at the same time when occupying the same channel [29]. A collision can alter part of the transmitted data, causing a checksum mismatch, causing the data to be invalid and ignored by the receiving node,[31] since the data packet ignored by the receiver is initiated for the affected packet. This can lead to exhaustion of the resources of the node by forcing the node to re-transmit data for every collided packet. Moreover, if this attack is launched on a large scale, it can lead to denial of service and exhaust the entire network [29].

2.3.2 Network

This layer of the IoT stack combines more than one feature, including routing, adaption, and fragmentation. Therefore, complex attacks can occur at this layer, from route manipulation to fragmentation, all of which can affect the availability of network resources. The attacks that affect network layer functionality are listed as follows:

Buffer Reservation Attack

This attack utilizes the fragmentation and reassembly functionality in the 6LoWPAN protocol. The core idea of this attack is to use a flaw in the buffer mechanism when handling fragmented packets. When the target node receives the first fragment of the packet, it reserves the entire buffer and waits for the other fragments to reassemble them. Attackers can utilize this flaw to send only one fragment and reserve the buffer for the maximum time allowed, which is defined by the 6LoWPAN protocol to around 60 seconds. Therefore, the intruder node can repeatedly send the first fragment to the receiving node and occupy the buffer for as long as it can, causing the victim node to be drained of all its resources [32].

Selective Forwarding

This is one of the popular routing attacks which tries selectively to forward only particular packets to the next node by dropping specific data of the packets. This

attack can be extremely dangerous when it is combined with other attacks like the sinkhole attack, which can lead to DoS.

Blackhole

As previously explained in chapter 1, the blackhole attack is similar in nature to the selective forwarding attack. However, instead of forwarding the packet maliciously it drops the entire packet.

Sinkhole

In a sinkhole attack, the malicious node advertises itself as having a better rank than the parent nodes, causing the neighboring nodes to change their parent and alter the routing path. The nearby nodes change their route to the sink because of the better fake route provided by the malicious node. A sinkhole might not be effective when it is executed by itself, but it can be far more critical when combined with other attacks like the blackhole or selective forwarding attack.

Sybil

A Sybil attack is an attack on node identity. It can have many forms, but it is a common identity fabrication attack where the advisory node tries to advertise itself as a different node in the network by stealing or fabricating another node's identity.

Flooding

In a flooding attack, the attacker node sends an unlimited number of DIS messages to the victim node over a short period, causing a DoS attack and disabling the node services. This kind of attack can also lead to battery exhaustion due to the consumption of node resources.

Wormhole

A wormhole attack is a routing-based attack where the attacker uses one or more nodes to create a fake tunnel with a better rank than the normal route to the sink node. Therefore, instead of data transmitting through the legitimate node, it uses

the fake tunnel to transmit data [33]. This attack can cause disturbances to network communications, including eavesdropping, selective dropping, and DoS attack.

TCP Hijacking

The transport mechanism in IoT uses either UDP or TCP protocols to transport data to the application layer. When using the TCP protocol, it inherits all the available flaws and vulnerabilities, one of which is top session hijacking [34] where the attacker tries to steal the client's identity because the attacker knows the sequence number and communication port. Later, the attacker can launch DoS attacks on the victim and assume its identity to communicate with the server.

6LoWPAN Fragmentation Attack

In IoT, when using the IEEE 802.15.4 standard, the user is limited to an MTU of 127 bytes using the 6LoWPAN fragmentation mechanism, which allows the transmission of IPv6/4 packets. The problem with 6LoWPAN is that it does not provide any kind of authentication, which means an attacker can inject their fragments among other fragments [11].

This next section of the research explores all the available methods used to counter DDoS attacks in IoT.

Table 2.4 : Protocol-based solutions

Study	Attacks	Layer	Method	Performance	Placement	Description
[35]	SH,SF,BH,SY	NTWRPL	Authentication / Encryption	H	C	hash chaining authentication approach.
[36]	SF,SH,HF,WH,CID	NTWRPL	Specification	L	C	Lightweight Heartbeat, RBL & IDS
[37]	MQ,RP	Cross Layers	Bio-metric based Encryption /authentication	∼M	C/D	Bio-metric Authetcation and encypayion.
[38]	*	NTWRPL	Authentication / Encryption		C	Hashing approach and private key
[39]	SF	NTWRPL	Authentication		C	Hashing and Map Function
[40]	SF,BH	NTWRPL	Data and route Duplication		C	data redundancy
[41]	FM	NTWRPL	Encryption		C	encrypted CGA-IPV6
[42]	FM	NTWRPL	Hashing		C	Hash chaining for secure RPL Rank
[43]	DI	NTWRPL	Encryption		C	encrypted chain

DD - DDoS attack, **Sh**-sinkhole attack, **SF**-Selective Forwarding , **BH**-Blackhole, **HF**-Hello Flood attack, **NTW/RPL** - Network Layer, **L**-Low,**M**-Medium , **H**-High, **D**- Distributed , **C**- Centralized, **U**-Undefined

2.4 Literature Review

In examining the literature, this study found the published studies can be categorized based on the type of solution proposed. Therefore, we divide the literature into

four categories: intrusion detection system (IDS)-based solutions, protocol-based solutions, trust-based solutions, and others.

2.4.1 Protocol-Based Solutions

Protocol-based solutions utilize the existing protocols to mitigate security flaws by enhancing the existing method or building new methods on top of the existing one. An example of such a solution is proposed [35], which investigated the use of a chaining message authentication code and the advanced encryption standard to cipher the packet payload between entities. The authors termed this framework 6lowPSec, and it works under the MAC security sub-layer in the adaption layer. They evaluated the system against many attacks, one of which was denial-of-service attacks, as the authors stated the solution was able to counter such malicious activities. However, the system's performance decreases when new nodes are added, causing the proposed model to take longer to process. [36] presented a comprehensive analysis of IoT technologies and their new security capabilities that can be exploited by attackers. One of the highlights is the implementation and demonstration of well-known routing attacks against 6LoWPAN networks running RPL as a routing protocol, which they simulated using the Cooja simulator and the Contiki operating system. The following RPL attacks were used for testing: selective-forwarding attacks, sinkhole attacks, hello flood attacks, and wormhole attacks. The testing results show that while the RPL protocol is vulnerable to different routing attacks, it has internal mechanisms to counter hello flood attacks and mitigate the effects of sinkhole attacks. The authors claim to implement a solution that minimizes selective forwarding attacks by implementing a heartbeat protocol on top of the IPSEC function in the ipv6 protocol. The basic idea is to send ICMPv6 messages from the 6BR router to all nodes in the network and wait for the ICMPv6 echo reply from the nodes. This technique has been implemented such that it sends ICMPv6 messages in an interval time, hence it is called the heartbeat protocol. The authors claim that this technique helps identify which node has been filtered using the IPsec protocol, hence identifying any node that may have been the victim of the attack. Bio-metrics play a significant role in security, but few studies have focused on IoT applications. The

work in [37] presents four layers of bio-metric architecture to provide an end-to-end solution for secure communication. The proposed architecture focuses on authenticating communication by using bio-metric devices and pairing-based cryptography to secure the data in transit. The core idea is to use a three-level interaction between layers to establish a communication channel between the layers. The system uses private key generator cryptography at each layer to ensure the secure transmission of biometric data. The authors claim that by encrypting the barometric data, the proposed protocol is resilient against masquerade and reply attacks. The problem with such a model is that it can introduce a communication overhead and heavy resource consumption on the end devices. Bio-metric solutions have a large data footprint compared to other authentication solutions such as encryption, and solutions can be challenging for devices with limited resources. Similarly, [38] proposed a new security protocol to secure RPL networks and called it (SRPL). It uses a hash chaining authentication approach to validate the authenticity of each node. SRPL has three stages: the first stage is the initiation phase, where the node calculates the hash and the threshold values when the DODAG is created. The second stage is the verification stage, where parents check the child node's validity by checking the hash and threshold values. The third stage is where the hash and threshold values are updated when any changes in the rank are signaled. To counter the selective forwarding attack, [39] proposed SCAD, a lightweight verification method that utilizes the map hash function that sends a frequent acknowledgment packet between the source node and the sink. To ensure secure communication between the source and destination, the author proposes placing a checkpoint node that piggybacks the connection packets with a unique random hashed number. This number serves as the ID between the checkpoint, source, and destination nodes. Furthermore, to reduce packet delivery latency caused by the attack, the authors propose a timeout technique based on estimated single-hop transmissions. One of the limitations of this study is that it depends on a static link between nodes. However, in practical, real-world IoT scenarios, nodes change their communication link dynamically. Therefore, this method is not practical in a dynamically changing topology.

A significant challenge for the RPL protocol is countering insider attacks with limited resource devices. To enhance RPL resilience against attacks [40] introduced a new method that uses randomized route selection and data duplication techniques. Duplicating the data and sending them through randomized parent nodes ensures that if one link is compromised, the data will reach the sink through the other randomized link. The author assumes that the IoT network is dense, and each node has multiple routing parents. Another study that focuses on solving one limitation of the RPL protocol was proposed by [42] to address the version number attack. The proposed method utilizes hash chains to authenticate the rank exchange between nodes. However, this method was later criticized by [43] as it is still susceptible to forgery and replay attacks. Therefore [43] proposed an enhancement to VeRA using an encryption chain instead of a MaC hash chain. The encryption chain for every node is calculated by the root/sink node. Also, the authors proposed a new method called TRAIL to authenticate the topology in the network. This method is used to prevent topology inconsistency in RPL networks.

Another method that uses cryptography was proposed by [41] to counter the 6LoWPAN fragmentation attack. Each joining node is assigned a temporary address by the border router (BR) and then selects its parent node based on its location in the network. To ensure safe communication, the method uses the ECQV implicit certificate-based cryptography, which is computed by the BR and assigned an encrypted CGA-ipv6 address, dropping the temporary address. The new address is used as a secure channel between the nodes in the network.

The method proposed by [44] is based on a statistical model where they use the sequential probability test [45] to estimate the dropped packet between the node and the sink. This is achieved by sending a hello packet using a dynamic adaptive threshold. If the dropped packet rate exceeds the predefined threshold, this indicates the node is malicious and will be blocked from the network. This method is used to detect a selective forwarding attack.

Discussion

In protocol-based solutions, few methods have explored the different aspects of integrating new protocols into the system. By thoroughly investigating the methods in the literature, we define the following limitations:

- **Cross-Layer:** as can be seen from Table 2.4, only one study has designed a solution addressing multiple layers attacks on IoT architecture. The study proposed by [37] used bio-metric solutions to provide a more secure architecture for data communication. However, the study fails to report the performance with respect to IoT limitations. The biometric data has a large footprint compared to encryption/authorization solutions. It might be more secure in terms of the uniqueness of biometric features, but we do not see this solution as feasible for constrained devices with the framework proposed. Other studies have not reported any cross-layer integration.

- **Evaluations:** most studies focus on simulation-based evaluation without introducing any real-world elements, such as noise and signal distribution objects, which can affect the result when deployed in real-world scenarios. The study proposed by [36] provides a comprehensive evaluation of the heartbeat protocol proposed, but has does not report how such solutions will perform in real-world scenarios [15].

- **Heterogeneity:** from Table 2.4 we can see that none of the studies addresses the heterogeneous nature of the IoT network, although the architecture biometric proposed by [37] is a cross-layer solution, but the authors do not explore the idea of supporting multiple technologies to address heterogeneity problems.

2.4.2 Trust Based Solution

IoT devices and networks are designed to create business value by connecting different kinds of devices and objects, no matter what resources are available in the end devices. This is why many devices with low memory and computing power are a part of this ecosystem. Therefore, when designing any trust management

Table 2.5 : Trust-based solutions summary

Study	Targeted Attacks	layers	Trust Measurement Method	Evaluation		Placement	Info Collection	
				Performance	Scalability		Indirect	Direct
[46]	SH	NTW/RPL	Trust-based	-	-	H/N	-	x
[47]	SF,SH,VN	NTW/RPL	Trust Value	-	x	H	-	-
[18]	BH	NTW/RPL	Trust	x	-	H/N	x	x
[48]	SY	NTW/RPL	Trust Based -IDS	-	-	H/N	-	x
[49]	SD	NTW/Any	Message based	x	x	H/N	-	-
[27]	SD	NTW/Any	Message based	x	x	H/N	-	-

DD - DDoS , **Sh**-sinkhole , **SF**-Selective Forwarding , **BH**-Blackhole, **HF**-Hello Flood, **NTW/RPL** - Network Layer, **L**-Low,**M**-Medium , **H**-High, **D**- Distributed , **C**- Centralized, **U**-Undefined

system (TMS), constraints and limitations related to such an ecosystem should be considered.

Many architectures for trust management are available in the literature. In this survey, we focus on the trust architecture that is based on a three-layer architecture [50]. The trust-based mechanism proposed by [47] is based on a trust value calculated using the subjective logic approach [51] and is evaluated using the opinion triangle (OP). OP evaluates trust-based on three attributes: trust, distrust, and uncertainty. In contrast to the traditional method where only two attributes are considered, this method explores the grey area where further analysis is required by using uncertainty attributes. To calculate trust, the authors assume the node is in a promiscuous mode, which allows them to hear a neighbor's node traffic. If the neighbor's node's trust value is low, this means that the node is distrustful, and the monitoring node will prohibit data from being transmitted through it. The authors suggested using this technique to counter the selective forwarding attack, sinkhole attack, and version number attack. The work done by [18] focuses on a trust-based solution to counter a blackhole attack in RPL networks. Essentially, a trust value for each node is calculated based on the number of packets sent and delivered through the parent node. The proposed mechanism also calculates the feedback value between nodes, which is the ratio of packets a node can successfully forward. Utilizing the feedback value, a blackhole attacker can be detected by monitoring the number of packets it dropped, hence giving a low feedback value. The proposed model has two assumptions: the first is that every node in the network will overhear their neighbor's nodes and the transmitted packets. The second assumption is that the

blackhole attacker will start dropping every packet it receives over time. This approach has some limitations, as it does not mention how the trust value is utilized to prevent a blackhole attack. Secondly, it assumes that all nodes are in promiscuous mode, which can minimize the lifespan of battery-powered devices.

Another study that utilizes the use of promiscuous nodes to detect blackhole attack is proposed by [49]. This technique utilizes the promiscuous nature of nodes to overhear the neighbor's node traffic and determine if the node is misbehaving or not. A local decision process uses a specific threshold to determine if any node is suspicious or not. To further investigate the suspicious node, a verification process is called, which uses two types of messages: the received Request (RREQ) and the received Result(RRES) messages. The RREQ is initiated by the verification node, and it carries a request asking the root node if the forwarding packet was received or not. The RREQ messages are sent through an alternative path not to be affected by the attacker node. The root then will send RRES, which refers to whether the forwarding message was received or not. If the message was not received, the attacker node will be flagged to the blacklist and broadcast to the whole network to avoid the attacker node. In this study, the author fails to mention how to calculate the misbehaving node, and there is not enough information about the misbehaving threshold. Moreover, the study does not mention what happens if there is no alternative route to the root node.

A study that tries to solve the problem of IoT heterogeneity was proposed by [27]. The authors proposed a context-aware trust management system that uses a dynamic trust score based on the node context and its status and proposes the use of different trust calculating functions for different node services. The system's centralized design may help reduce network overhead but can lead to a single point of failure if the system fails. However, the author does not explain how the system will scale in a large dense network [52].

Discussion

Based on the aforementioned studies, Table 2.5 provides a comprehensive summary of the explored trust studies. As can be seen, most of the studies affect the network layer, referred to as communication trust in the trust management framework. The observation in this context is that the trust evaluation schema in each study is limited, but most of the studies fail to address the limitations associated with IoT devices.

IoT Limitations: Although some of the proposed solutions achieve excellent results in terms of trustworthiness and accuracy, they fail to adapt to IoT limitations and constraints, since these solutions require an extensive amount of CPU and memory power which is not applicable in the context of limited-resource devices. At a glance, these studies [48] [28] appear to have good results in terms of trust accuracy; however, they fail to report the system solution from the IoT device's perspective.

Cross Layer: As shown in the summary table, most of the studies focus on communication layer trust-based solutions and ignore multiple-layer adaptation. All of the studies reviewed focus on one-layer solution and ignore the trust issues that appear at a different layer of the ecosystem. Designing a cross-layer solution is crucial to handle security breaches at a different layer of the IoT architecture.

This shows that the literature lacks a reliable and scalable trust management framework that can consider the limitations associated with IoT networks.

Evaluation: The studies focused mostly on simulation-based evaluation without considering real-world elements, such as noise, which is usually an essential factor when deploying a solution in the real world.

2.4.3 Intrusion Detection Based Solutions

The intrusion detection system (IDS) has been used for some time in different network applications. The main purpose of IDS is to detect any suspicious activity against the targeted network. There are various approaches in IDS, which can be

Table 2.6 : Intrusion detection solution summary

Study	Attacks	Layer	Method	Performance	Placement	Brief	Dataset
[13]	DD	NTW/RPL	Signature-based	L	D	Complicated DOS attacks,	KDD
[53]	SH	NTW/RPL	Specification	L	C	Leader Node Selection,	KDD
[54]	SH	NTW/RPL	Specification	L	C	semi-auto profiling technique	KDD
[55]	SH,SF,BH	NTW/RPL	Hybrid(Specification / Signature)	L	C	Fixing network inconsistency	KDD
[56]	SF	NTW/RPL	IDS	M	D	Deep packet Inspection , Misbehavior	KDD
[57]	SF	NTW/RPL	Anomaly	M	D	Anomaly Based solution	KDD
[58]	HF,SY	NTW/RPL	Encryption	U	D	two ray prorogation model	KDD
[59]	DD	NTW/RPL	Specification	L	D	Identify DDoS attacks before targeting network,	KDD
[60]	HF,SM	NTW/RPL	IDS	*	D	knowledge-drive IDS	KDD
[61]	HF,BH	NTW/RPL	Authentication	U	D	Signature-based	KDD
[62]	SH	NTWr/RPL	Packet Inspection	H	D	Based on IR value, packet Received , Packet Sent	KDD
[63]	SH,SF	NTWr/RPL	Specification& Anomaly H	D	packet Received , Packet Sent	-	

DD - DDoS , **Sh**-sinkhole , **SF**-Selective Forwarding , **BH**-Blackhole, **HF**-Hello Flood , **NTW/RPL** - Network Layer, **L**-Low,**M**-Medium , **H**-High, **D**- Distributed , **C**-Centralized, **U**-Undefined

classified into four categories:

- Signature-based approach: The system detects an attack by comparing the signature of the activity against a pre-installed set of signatures in the IDS database. If there is a mismatch in the signature, the system raises the alarm.
- Anomaly-based approach: In this approach, the IDS is trained to detect any anomalies in the network by analyzing their behavior, and if any activity exceeds a specific threshold, this indicates that an attack has happened.
- Specification-based approach: In a specification-based approach, the IDS checks network activity against a set of predefined rules and settings. This approach detects misbehaving intruders when their activity does not have the same specifications as defined in the system. This approach is sometimes called the rule-based approach.
- The hybrid-based approach combines more than one approach to maximize the advantages of each and minimize the drawbacks.

An excellent example of a signature-based IDS is the architecture proposed by [13]. It integrates an IDS into a network that has been developed within the EU FP7 project Ebbits. The goal of this proposed architecture is to detect threats on a 6LoWPAN network. The authors studied and analyzed DOS attacks in IP-based WSNs and proposed a solution involving a signature-based IDS that uses a predefined source of signatures and patterns collected before implementing the solution. The authors used probes in the edge of the network to sniff packets that

go through the entire network and analyze each packet to look for any suspicious behavior, which is later sent to SenacIntrture IDS for further analysis.

Likewise, [64] uses the same approach, which is dependent on the specification-based approach, but the main focus of their study is to build an IDS that addresses routing attacks in an RPL network using a semi-auto profiling technique. This technique was used to gather and formulate a set of rules that is integrated into the IDS agent. The placement of the IDS was chosen carefully by the authors to eliminate any kind of network resource overuse. Therefore, they placed the IDS as a cluster head agent as they assume the network is cluster-based.

[55] uses a hybrid technology combining the signature-based approach and anomaly-detection approach. This approach utilizes the consumption of the limited resources of the signature-based approach and when combined with the accuracy of the anomaly detection technique, it produces better results. The basic idea of Svelte is to implement the IDS in a distributed approach, which is later installed across every node and implemented in the 6BR router. To fix network inconsistency, the authors developed a 6Mapper on top of the RPL protocol. The primary function of the 6Mapper is to fix the network inconsistency caused by either a hacker from within the network who sends incorrect information to its neighbors or the loss nature of an IoT network, which can cause inconsistency. [55] used the Contiki OS RPL implementation to develop the 6Mapper on top of the system.

In addition to the 6Mapper, the authors developed a mini-firewall to detect any global attacks coming from outside by distributing the mini firewall across all the constrained nodes in the network, with the main module installed in the 6BR router. The authors claim that this helps minimize the overhead in the network.

[56] proposed a monitoring tool for attack inspection and detection, which uses a deep packet inspection approach to investigate network traffic and identifies any misbehavior based on a set of rules defined in an XML file.

In the method proposed by [65], fog computing is used to counter selective forwarding attacks in sensing networks. The core idea is to build an intrusion detection

system in fog computers at the edge of the network. The author proposed the use of watchdog nodes that monitor any suspicious node while on the move. The watchdog nodes maintain the received packet and the sent packet by the monitored node. These values are forwarded to the fog node for further processing, where the fog node decides whether the monitored node is malicious or not, based on a specific threshold. Unfortunately, the authors do not mention how to calculate the threshold, and the study lacks details about the approach.

The "Kalis" IDS architecture proposed by [60] utilizes a knowledge-driven intrusion detection system to counter hello flood and smurf attacks, where the IDS observes the network traffic to extract specific features and feeds them to what is called knowledge-base storage. Using the knowledge gained about the specific node, the system identifies the malicious node and triggers the specific detection mechanism. However, the study does not explore how the feature process is executed and what features are collected for the knowledge base database.

Another hybrid method that uses an anomaly and specification detection mechanism is proposed by [63]. The placement of this IDS is distributed between the router and the sink node in the network, where the specification-based agent works as a general inspection tool for all of the nodes in the network. In the case of a suspected attack, the router forwards this information to the sink node for further processing. Using anomaly-based detection, the root node extracts specific features from the communication data and analyzes them for any malicious activity. The process is then passed to a voting system that learns by analyzing the network behaviors. This IDS is used to counter selective forwarding and sinkhole attack.

Discussion

Based on the aforementioned literature in the context of IDSs in IoT, the following observations can be made:

IDS Methods: Although there are variations of IDS placement choice in the literature, most do not identify the pros and cons of choosing such placements. The hybrid method has shown excellent results when it comes to attacking accuracy

and fast response, but none of the studies have thoroughly investigated the scalability and performance of such methods in real-world scenarios. By examining the literature, we find that the only hybrid IDS that provides enough details for an evaluation and testing scenario is proposed by [55] as it tries to address the limitations introduced by specification and anomaly detection when they work separately. One weakness of the proposed system is that it does not provide any details on how the system will evaluate different kinds of attacks or protocols, and although the author claims there is a possibility of expanding the system, no detailed information is provided.

IDS Placement: There are three types of IDS placements: distributed, where the IDS is installed across the network; centralized, where all data processing and attack detection happens on a single node that has more resources then other nodes in the network, and hybrid, where it tries to overcome the limitations of the centralized and distributed approach by organizing the network into a group of clusters. Each cluster has a root node that interacts with the main IDS component, usually installed in the 6BR router. Although there are various placement strategies, most of these studies fail to point out the performance trade-off of each placement. [46] presented how distributed placement methods can help detect sinkhole attacks more efficiently, but the authors offer no details about how the placement of the IDS helps in achieving excellent results in attack detection.

IDS Heterogeneity problem: In the literature, most of the studies discussed the 6LoWPAN protocol, and building an IDS on top of the 6LoWPAN protocols is often proposed. This can be easily explained due to the standardization by the ITTF organization, where most IoT manufacturing companies adopt the 6LoWPAN and the RPL protocol. A problem arises when the IoT network is combined with different kinds of protocols and technologies like the Z-Wave, ZigBee, and BLE, as this can cause a miscommunication problem that needs to be addressed when designing any IDS. The heterogeneity nature of IoT networks allows different manufacturers and organizations to form the IoT network; therefore, designing a solution that considers this aspect is crucial.

IDS Targeted Attacks: None of the explored studies focus on realizing the concept of attack detection. Most of the studies concentrate on particular attacks like blackhole attacks or sinkhole attacks. However, most of these attacks can be combined to have a more disastrous effect. None of the studies explored in this literature review investigated the idea of integrated attacks that work collectively to affect the system maliciously. Another limitation observed in the literature is the limited number of studies that focus on physical and application-layer attacks. The majority of the studies retrieved focused more on attacks on the network and adaptation layer. A cross-layer solution will help address this limitation.

IDS Dataset limitation: As can be seen from the summary presented in Table 2.6 , all of the studies used the DARPA, and KDD datasets to evaluate and test the proposed solutions. The limitations of the DARPA and the KDD datasets, namely their biased nature, are many duplicated records, they are outdated, and they are not explicitly designed for use in IoT networks. Moreover, these datasets use a different set of protocols and attack emulation that are not supported by most IoT network architectures.

2.4.4 Others

[66] proposed a technique that combines the return-oriented programming ROP approach with code checking to provide extra security against malicious code tampering in IoT devices. It aims to protect the most critical part of the code by including a two-level module (ROP and Checksumming) in the program's tamper-resistance section. Such a technique adds an extra level of difficulty in terms of a tampering attack since the attacker has to bypass two modules to establish a successful attack, therefore increasing the cost of exploitation. This approach has some limitations from the user access perspective, although the authors stated no additional performance is compromised when using these techniques. The authors do not state how the approach will affect battery-powered devices (since most IoT devices are battery powered).

Another method that utilizes the use of the ROP technique is presented by [67].

The authors proposed a model that assumes a hostile user has access privileges to control the program's entire run environment. The proposed model uses the Genetic Algorithm to choose the best devices with the minimum execution time to optimize the ROP chain.

[68] introduced an anti-jamming approach for OFDM-based IoT devices by utilizing a game-theory technique. The proposed approach uses the Colonel Blotto game to establish an interaction between the jammer and the IoT controller. The IoT controller defends the network against jamming attacks by intelligently distributing the attack power across sub-carriers, which causes the bit error rate (BER) to decrease, reducing the effect of the attack. Through simulation, the authors demonstrate the effectiveness of such an approach in maintaining healthy network performance and a good BER.

Similarly, [69] explored the hierarchical security game approach to form a competitive relationship that tricks the jammer into taking action after the legitimate node starts transmission to stop what is called reactive jamming. By utilizing such techniques, the study forces the victim node to take action first, hence minimizing the effect of the jamming attack. To achieve this level of protection, the authors suggest that the legitimate user determines its transmitting power (since the attacker uses the transmitting power to launch the attack) to trade-off between the signal-to-noise ratio and the probability of being detected and jammed.

Discussion:

The method proposed in this section varies between detecting tampering and jamming attacks. Although some might consider these attacks outside the scope of this survey, we have found that these attacks can serve as the starting point to launch complicated attacks that affect the availability of the network. The study by [66] uses the ROP code checksumming method to protect IoT devices from code tampering. However, the study does not mention how to implement this solution on different platforms to adapt to the IoT network's heterogeneity. Another study that uses the ROP method was proposed by [67], but it might not apply to many IoT

Table 2.7 : Overview of IoT security approaches

Study	Solution	Attacks	Performance				Evaluation		Scalability	Heterogeneity	New Data	Attacks	IoT
			E	C	N	M	S	R					
[35]	Protocol	SH,SF,BH,SY	x	x		-	x	-	x	-	-	-	-
[36]	Protocol	SF,SH,HF,WH,CID	x	x	-	-	-	-	x	-	-	x	-
[37]	Protocol	MQ,RP	-	-		-	-	x	-	x	x	x	-
[38]	Protocol	DD	-	-	-	-	-	-	-	-	-	-	
[39]	Protocol	SF	-	-	-	-	-	-		-	-	-	-
[40]	Protocol	MQ,RP	-	-		-	-	x	-	x	x	x	-
[41]	Protocol	FM	-	-		-	-	x	-	x	x	x	-
[44]	Protocol	SF	-	-		-	-	x	-	x	-	-	-
[42]	Protocol	VN	-	-		-	-	x	-	x	-	-	-
[43]	Protocol	VN,DI	-	-		-	-	x-	-	x	-	-	-
[18]	Trust	SH	-	-	x	-	-	-	x	-	x	-	-
[47]	Trust	SD	-	-		-	-	-	x	-	-	x	x
[49]	Trust	SD	x	x	x	x	-	-	x	-	x	-	x
[27]	Trust	*	x	x	x	x	-	-	x	-	x	-	x
[13]	IDS	DD,UDP flooding	-	-	x	-	-	x	-	-	-	-	-
[54]	IDS	SH	x	-	-	-	-	-	x	-	x	-	x
[55]	IDS	SH,SF,BH	x	-	x	x	x	-	x	-	-	-	x
[56]	IDS	*	-	-	x	-	-	-	x	-	x	-	-
[65]	IDS	SF	-	x	x	-	-	-	x	-	-	-	-
[60]	IDS	HF,SM	-	x	-	x	-	x	-	-	x	-	-
[62]	IDS	SH,SF	-	-	-	-	-	-	-	-	-	-	-
[63]	IDS	SH,SF	-	-	-	-	-	-	-	-	-	-	-
[68]	Other	JM	x	x	-	-	x	2	-	x	-	-	-
[66]	Other	JM	-	x	-	-	-	2	-	-	-	-	-
[69]	Other	TM	-	-	-	-	-	-	x	-	-	-	-
[67]	Other	TM	-	x	-	-	-	-	-	-	-	-	-

- Unsupported , **x** - Supported, **E**-Energy, **C**-cpu , **N**-Network , **M**-Memory , **S**-Simulation , **R**-Real-world Scenario , **DD** - DDoS , **Sh**-sinkhole , **SF**-Selective Forwarding , **BH**-Blackhole , **HF**-Hello Flood , **DI**-DoDAG inconsistency , **SD**-Sleep Deprivation , **CID** -Clone ID , **MQ**-Masquerade , **WH**-Wormhole , **SM**-Smurf , **VN**-Version Number , **JM**- Jamming , **TM** -Tampering , **P**- Partially

architectures since it assumes the user has access permission to control the entire run environment. This can limit the application of such techniques.

2.5 Comprehensive Discussion

After examining the literature, we have created a comprehensive table examining each proposed solution, and their limitations and the following issues are identified:

- **Limited Evaluation Parameters:** As can be seen from Table 2.7, only a limited number of studies cover the four aspects of the evaluation process. When designing any solution for IoT, it is important to consider these parameters. For example, network overhead can cause the system to drain the energy resources of the IoT device. Therefore, analyzing every aspect of the system from the perspective of energy, computing, and network resources is

essential when designing a security solution for IoT. Although some studies have good accuracy in their attack detection results, they, unfortunately, fail to report the performance evaluation from resource perspectives, such as the solution proposed by [55] and the solution proposed by [67].

- **Scalability :** Scalability is crucial when it comes to measuring how the new system performs in a large dynamic network, where nodes frequently join and leave the network. To minimize performance degradation when new nodes are added to the network and to increase the scalability of the system, several studies show that grouping nodes into clusters can help in some scenarios [70]. To address the scalability issues, [60] proposed the addition of new IDS nodes across the network. The more extensive the network, the more IDS nodes the network will require. Unfortunately, most of the studies in this survey do not explain how their system scales when a large number of nodes are introduced into the network.

- **Heterogeneity:** IoT networks are different from the others because of the heterogeneous nature of the network. IoT networks comprise different devices from different manufacturers running various applications and operating systems. Under this level of complexity, the security solutions should be interoperable in such ecosystems. The study by [37], addresses this issue. However, limited information exists on how the system functions under different environments. Other studies reviewed in this chapter do not address this issue, whereas most of the proposed systems focus on a particular set of protocols and environments.

- **Datasets:** Anomaly detection IDS requires a good dataset to produce a good unbiased result in the training and testing phase. Therefore, choosing the right dataset for the system is crucial. [37, 71], used the KDD dataset [72]. However, as discussed previously, it is an outdated dataset and has been criticized on many occasions [73] [74] [75]. Furthermore, the KDD and DArapa datasets were created in a very different environment and with different protocols to those used in IoT networks today. Therefore, building a dataset that uses

appropriate protocols and architecture is a vital element in building a reliable framework for any anomaly detection solution. A common protocol used in the IoT network is RPL for routing, 6LoWPAN for adaption, and MQTT and COAP for a top-layer application interface. Hence, creating a dataset with these protocols is crucial to producing an accurate and relevant result.

- **Multi layer solution:** Another critical observation from Table 2.7 is that limited studies focus on building a solution that covers more than one layer of the IoT architecture. The only study that proposes a multi-layer solution is the one proposed by [37], where the authors use a biometric solution on a different layer. Although, in theory, this can increase attack prevention, it is not practical due to the network overhead and performance. Most solutions focus on network and topology layer attacks, such as sinkhole and blackhole attacks, as shown in Figure 2.3. What the solution distribution shows is that the majority of research has focused on the network and physical layer, with less research conducted on top layer attacks such as COAP and MQTT-related denial-of-services attacks. Therefore, there is a vast research opportunity in this specific area.

2.5.1 Summary of the Literature Shortcoming

From this comprehensive discussion based on the literature review, we can summarize the literature's limitation as follows:

- All the studies focus on traditional specifications and anomaly detection without exploring the potential of integrating machine learning algorithms to solutions.
- None of the existing literature explore the use of machine learning to detect selective and black hole attacks in IoT.
- None of the existing literature explore the use of machine learning classifiers to detect UDP flooding attacks in IoT.

- The existing research also lacks the idea of developing a traffic aggregation tool to build datasets to be used in for machine learning IDS in the IoT environment.

- No dataset was explicitly designed for IoT attack detection. Most of the solutions proposed used, similar to the KDD dataset are outdated.

2.6 Conclusion

Security is a crucial element in determining how the IoT network and devices will help shape our future. One of IoT's most significant security challenges is how to detect DDoS attacks without compromising the limitations associated with IoT devices. This systematic literature review has presented a comprehensive survey of the current DDoS attack detection approach, and we have identified the attacks associated with DDoS that affect the availability of the network. Furthermore, in addition to exploring the limitations associated with the detection approach, we have identified aspects that should be considered when designing IoT security solutions. We evaluated the proposed solutions to DDoS attacks in terms of practicality in real-world scenarios. In the next chapter, the limitation derived from this systematic review will be used to form the research questions and objectives.

Chapter 3

Problem Definition

3.1 Introduction

As discussed in the previous chapters, IoT security is a challenging task due to a variety of factors, from the heterogeneity of the IoT network to the limitations associated with limited resource devices. Building a solution for such a network requires careful consideration of its constraints and limitations. For example, a traditional cryptography solution which requires high computational power will not perform efficiently in the IoT environment due to the limited memory and CPU power required by IoT devices. Therefore, depending on traditional methods without modification can lead to failure in most scenarios. Many studies and solutions have been proposed in the literature, as was explored in Chapter 2. These solutions used different techniques and approaches, but many failed to address the problem of heterogeneity and performance problems. Moreover, none of the existing studies explored the use of machine learning to address DDoS attacks in IoT networks. Nevertheless, most of the studies that have been done in similar networks such as MANETs and ad-hoc networks used unreliable datasets such as the KDD dataset which is not appropriate for limited resource networks.

In order to address the above research gaps, in this study, we propose a robust solution to address DDoS attacks in the IoT network utilizing machine learning, which is the scope of this study. Chapter 2 systematically explored the literature and identified some gaps and limitations in the literature. These shortcomings are described in Section 3.3.

3.2 Terms

3.2.1 Internet of Things

The Internet of Things is the process of allowing things to be connected to the Internet. These things can include sensors and actuators embedded in the physical object and connected to the Internet through a wired or wireless connection [76].

3.2.2 IoT Security

We define IoT security as the process of protecting the IoT network and devices from any malicious behavior that can have negative consequences.

3.2.3 DDoS Attacks

we define DDoS attacks are the process of causing security damage to the network or the devices affecting the availability of the services either of the network or the devices.

3.2.4 Selective Forwarding Attacks

Selective forwarding attacks are a special kind of attack that affects the network's availability by disturbing the routing behavior in the network [77].

3.2.5 Blackhole Attack

Blackhole is a special variant of the selective forwarding attack which manipulates the node ranking by making the malicious node has higher ranking then the adjusnt nodes. Therefore every packet will be sent through the attacker node which will drop every packet it receives [78].

3.2.6 Flooding Attack

We define flooding attacks are the process of attacking the IoT network or device by sending a large number of unrelated messages to disrupt the network performance and cause denial of service.

3.2.7 6LoWPAN Protocol

The 6LoWPAN protocol is standardized by the ITTF to adapt a smaller version of the IPv6 protocol for a limited resource network like the IoT network. Moreover, it works as the middle-ware between the Internet and the IoT network [79].

3.2.8 RPL Protocol

The RPL protocol is standardized by the ITFF to work as IPv6 Routing Protocol for Low-Power and Lossy Networks [10].

3.2.9 SVM

One class SVM is a commonly used method that has proven its reliability and accuracy for different applications. The one class SVM is a special kind of SVM algorithm that classifies objects from one class only. The one class SVM is chosen in this research to look at the possibilities of what will happen if we do not have any attack data within our dataset; one of the advantages of one-class SVM is that it does not require attack data or anomalies in the dataset [80].

3.2.10 Neural Networks

Adaptive resonance theory (ART) is one of the popular models for neural networks. It adapts methods in relation to how the brain interacts and processes information [81].

3.2.11 Decision Tree

The decision tree is a machine-learning algorithm that is very widely used in the machine-learning community, specifically in the area of data mining [82].

3.2.12 Node

We define a node as a physical device or thing in the IoT network, ranging from simple sensor devices to very complicated actuators in industrial applications. A node is capable of receiving messages from and sending information to other nodes in the network.

3.2.13 Sink/Root Node

We define sink Node as the main node in the IoT network hierarchy. It stores information about other nodes in the network, such as routing information and the ranking for each node.

3.2.14 Sniffer Node

We define a sniffer node as a node in the network whose primary purpose is to listen to all neighboring node traffic within its range.

3.2.15 Attacker Node

We define an attacker node as the node that maliciously causes harm to the IoT network/device.

3.2.16 6BR Router

6BR router is the bridge between the IoT network and the Internet [79].

3.2.17 IDS

We define IDS as an intrusion detection system whose main purpose is to detect an attack by continuously analyzing the data it receives from the network.

3.2.18 IDS Agent

We define an IDS agent as small client software that syncs information with the main IDS. Usually, this agent is installed either on the sink node or the normal node.

3.2.19 Data Extraction

We define data extraction as the process of extracting precise information from the traffic transmitted in the network.

3.2.20 Feature Selection

We define feature selection as the process of selecting the best feature associated with DDoS detection, which will later be added to the detection model to give the

best result possible.

3.2.21 DODAG

The direction-oriented directed acyclic graph (DODAG) is a tree-like hierarchical model used to build a path to the sensor nodes, ensuring that every node has a path and the root node can reach it [83].

3.2.22 DIS

DODAG information solicitation are special messages sent by the node to join the network [10].

3.2.23 DIO

DODAG information object is a core type of message of the RPL protocol which contains the necessary information to form the DODAG tree [10].

3.2.24 Route

We define this as the route from the sink node to any node in the network or vice versa. The relevance of the route to the others and its importance can be defined using the objective function as one of the parameters of the RPL protocol.

3.3 Problem Overview

As previously stated in earlier chapters, the need for an IoT DDoS detection solution that does not compromise the limitations and constraints associated with the IoT network is crucial. These limitations and constraints include limited resource devices, privacy, authentication, services and the data management challenge. Chapter 2 comprehensively investigated the available solutions in the context of detecting DDoS attacks in IoT. Most studies in the literature focus on adapting traditional network solutions into limited resource networks, therefore failing to address the problem of IoT heterogeneity and resource limitation. Nevertheless, although some studies were able to pinpoint the issues related to DDoS attacks in IoT networks,

they failed to suggest or propose robust and practical solutions to counter such problems. Some studies proposed the use of cryptography to ensure safe and secure communication between the nodes. However, applying cryptography solutions can cause an increase in traffic overhead and an increase in resource consumption in the node due to high computation demands.

In the scope of this research, we focus on detecting DDoS attacks using machine learning. As we explored in the literature, machine learning has been used in different security applications due to its high precision and the learning nature of such an approach. It has been widely used in different network architectures for attack detection. Networks similar to IoT, like WSN, Ad-hoc and MANET networks use ML for different security applications such as anomaly detection and attack detection. Nevertheless, these applications adapt traditional network characteristics where it is, in most cases, not feasible for limited resource networks. Another critical limitation in the literature regarding machine learning applications is the use of the KDD dataset, an outdated dataset created for a different kind of network and different sets of protocols, which does not apply to resource-constrained networks.

Based on the problem defined above and the comprehensive literature review in Chapter 2, the research issues are defined in Section 3.4.

3.4 Research Issues

From the above comprehensive and thorough problem definition, we identify the following research issues in the IoT security literature:

1. There is no existing machine-learning method for handling DDoS attacks in IoT networks in the existing literature.

2. No work has been done on developing a data collection tool to build a dataset that can be used for machine learning in IoT. Most of the research uses the KDD dataset for training, an outdated dataset for emerging technologies.

3. Most of the literature presents conventional methods such as specification-based and signature-based methods to handle threats in IoT. None of the

existing literature focuses on the integration of machine learning methods for DDoS detection in IoT.

4. There is no work that examines the use of support vector machines (SVMs), neural networks, and decision trees to detect DDoS attacks in IoT.

5. There is no current work on developing preventive measures against blackhole and selective forwarding attacks in IoT networks using machine-learning methods.

3.5 Research Questions

Based on the gaps identified above and the research issues, the main research question of this project is as follows:

"How can DDoS attacks be accurately identified in IoT environments using machine-learning methods?"

This question can be further divided into four sub questions:

- **Research Question 1:**

  How to collect and analyze simulated IoT traffic data and build a training dataset (IoT-DDoS)?

- **Research Question 2:**

  How to evaluate and choose the best machine-learning approach for DDoS (selective forwarding, blackhole, and UDP flooding attacks) detection?

- **Research Question 3:** How to implement the selected machine-learning method in an emulated IoT environment?

- **Research Question 4:**

How to evaluate and benchmark the developed IDS using the KDD and IoT-DDoS dataset and compare the two datasets?

3.6 Research Objectives

The research objectives of this research are as follows:

- **Research Objective 1**

  To develop a systematic way to collect IoT communication data (IoT-DDoS) for DDoS evaluation and benchmarking.

- **Research Objective 2**

  To develop a machine learning framework to evaluate three machine learning methods using the IoT-DDoS dataset collected in objective 1.

- **Research Objective 3**

  To develop an intelligent IDS framework that uses the best machine learning model developed in objective 2 in an emulated IoT environment.

- **Research Objective 4**

  To evaluate the developed IDS using an IoT-DDoS dataset and various IoT network scenarios.

3.7 Research Approach to Problem-Solving

To address the aforementioned research questions and limitations in the previous literature, this thesis aims at implementing a state-of-the-art framework to intelligently detect DDoS attack in 6LoWPAN IoT networks. To solve the issues identified in the previous section, a systematic and scientific method must be followed. Generally, there are a couple of approaches to solving any scientific problem; the main ones are the design science approach and the social science approach which are described further as follows:

3.7.1 Design Science Research Methodology

In the design science approach, comprehensive observations of a problem in the computer science and engineering sector are made to create and evaluate new artifacts about the specific problem. In many cases, these artifacts are produced from

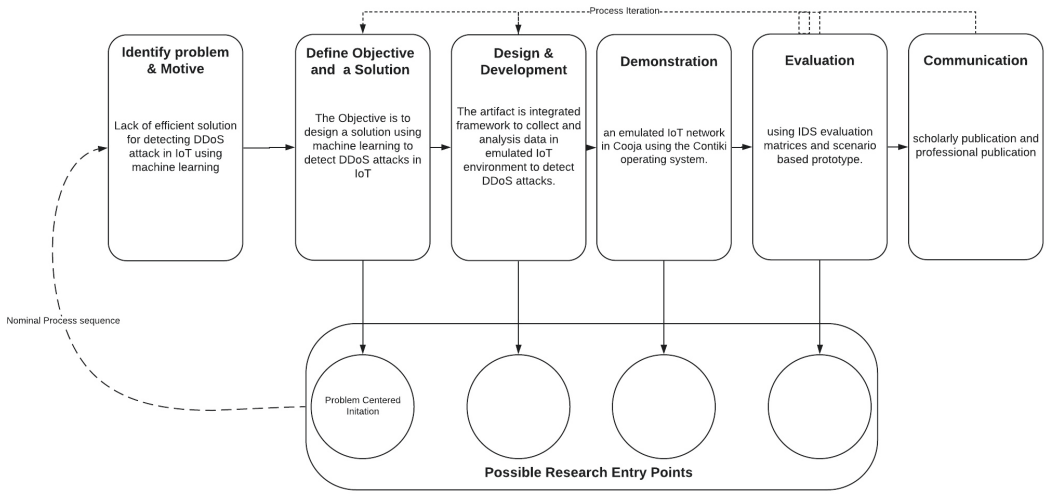


Figure 3.1 : Overview of the design science research methodology

existing knowledge within the literature. However, in some cases, further research is needed to address the limitations and gaps in the literature, which in this case, is considered innovative, and as a result, can be published or patented. There are six phases for the design science approaches, each of which are explained below as defined by Peffers [84] :

1. Problem identification and motivation: In this phase, the research problem is defined precisely, which will help in developing an artifact of the problem. Producing artifacts allows the researcher to develop an effective solution to the problem. The solution should be well justified to provide enough significance for the research and show how the researcher understands the scope of the problem and its complexity.

2. Define the objective and solution: In this phase, the objectives of the identified problem. The objectives should reflect the scope of the problem in terms of feasibility and achievability. The objective can be qualitative where, for example, the objective provides detailed knowledge of how the new artifact supports the proposed solution. On the other hand, the objectives can be quantitative, where they provide measurable values, for example, measuring

the performance of a system in a better way than has been proposed in the literature.

3. Design and development: In this phase, the artifacts are created, which can be any designed object that can be considered as a contribution. The designed artifacts should be well defined through a comprehensive iterative evaluation process.

4. Demonstration: In this phase, the researcher uses the identified artifacts to solve the problem. Knowledge of the artifacts and how to use them to solve the problem is required. This might involve scholarly activities such as an experiment, simulation, or a case study to show how the artifacts are used to solve the problem and how the artifact is relevant to the problem itself.

5. Evaluation: In this phase, an iterative process is carried out to measure and evaluate how the artifact is performing in support of the solution. To ensure the relevance of the results, the researcher should have extensive knowledge of the matrices used for the evaluation with a good knowledge of the observed result and comparing them with the achieved objective. At this stage, the process can be iterative until the objectives are met.

6. Communication: In this phase, the researcher focuses on communicating the result and the novelty of the solution provided and its significance to researchers and the professional community in that specific field. For people in academia, this may include paper publications and conference presentations following the empirical research design, which translates to the same stages being followed in this method.

3.7.2 Social Science Method

In using the social science research approach, the researcher focuses on following a systematic method to either disprove or prove a hypothesis based on a gathered knowledge in the form of data. The data collection process can involve interviews

or surveys [85]. This approach is divided into two categories which are explained as follows:

- **Quantitative approach:** factual data are collected that have numerical values which are usually statistical and structured [86]. In quantitative research, data can be measured and quantified.
- **Qualitative approach:** in qualitative research, the aim is to observe behaviors and phenomena and the results are not measurable; rather, they are described.

The social science approach assists in demystifying a social or cultural problem and provides a more informative explanation of the problem without proposing a method to solve it. On the other hand, the design science approach provides objectives and artifacts to solve the identified problem.

Hence, this thesis falls into the design science approach category, therefore providing a solution to the problem.

3.7.3 The Choice of the Science and Engineering-Based Research Method

This thesis follows the designs science approach to achieve the objective defined in Section 3.6. To understand how this thesis relates to the design and science approach, each phase of our research is explained as follows:

- Problem identifications: The research problem for this thesis is: "How to detect DDoS attacks in IoT using machine learning?"
- Define objective and solutions: Develop a data collection tool to generate a new dataset for IoT. In addition to this, a new IDS using machine learning is developed which utilizes the generated dataset.
- Design & development: Machine learning IDS for DDoS detection in IoT.
- Design & development: Demonstration is undertaken in a proof of concept simulation environment.

- Evaluation: using machine learning metrics / and comparing it to existing IDS methods in IoT (Svelte)
- Communication: Two conference papers two journal papers will be submitted to top quality publications.

3.8 Conclusion

This chapter has described the gaps and limitations in the literature related to DDoS detection in IoT. The terms and concepts used to design, implement, and evaluate the proposed framework are defined in this chapter in section 3.2. Furthermore, a precise definition of the problem is identified in relation to this thesis. To further investigate the defined problem, it was divided into four research questions. Finally, the research methodology used is described and detailed, corresponding to our research objective.

In the next chapter, a brief overview of the proposed framework and the solution for the related research questions is explained.

Chapter 4

Solution Overview

4.1 Introduction

The IoT network concept has evolved in the last few years to include many applications that had previously never been considered, such as smart factories with autonomous machinery and precision agriculture, where IoT devices are utilized to provide better crop cultivation and production capabilities. However, due to the complexity of IoT and its related technologies, IoT suffers from multiple challenges. One of these challenges is security, in particular, security issues related to network availability, such as DDoS attacks. To solve these issues presented in Chapter 3 of this thesis, this chapter presents an overview of the proposed solution to the problem of DDoS attack in the IoT network. The rest of this chapter is organized as follows; in Section 4.2, a brief explanation of the proposed machine learning framework is presented. Section 4.3 explores the data collection process using the proposed framework and network design. In Section 4.4, the machine learning methods, and the selection process with the evaluation metrics are outlined. In Section 4.5 an overview of the machine learning IDS implementation in the proposed network is presented. Finally the chapter concludes in Section 4.6.

4.2 Overview of IDD-IoT Framework Solution for DDoS Detection in IoT

In this part of the thesis, we present an intelligent framework for detecting DDoS attacks in IoT networks. For the rest of the thesis, the proposed framework is termed IDD-IoT. The IDD-IoT framework is an intelligent framework that utilizes a powerful machine learning algorithm for attack detection. The proposed framework comprises two phases, the pre-learning phase and the post-learning phase, as shown

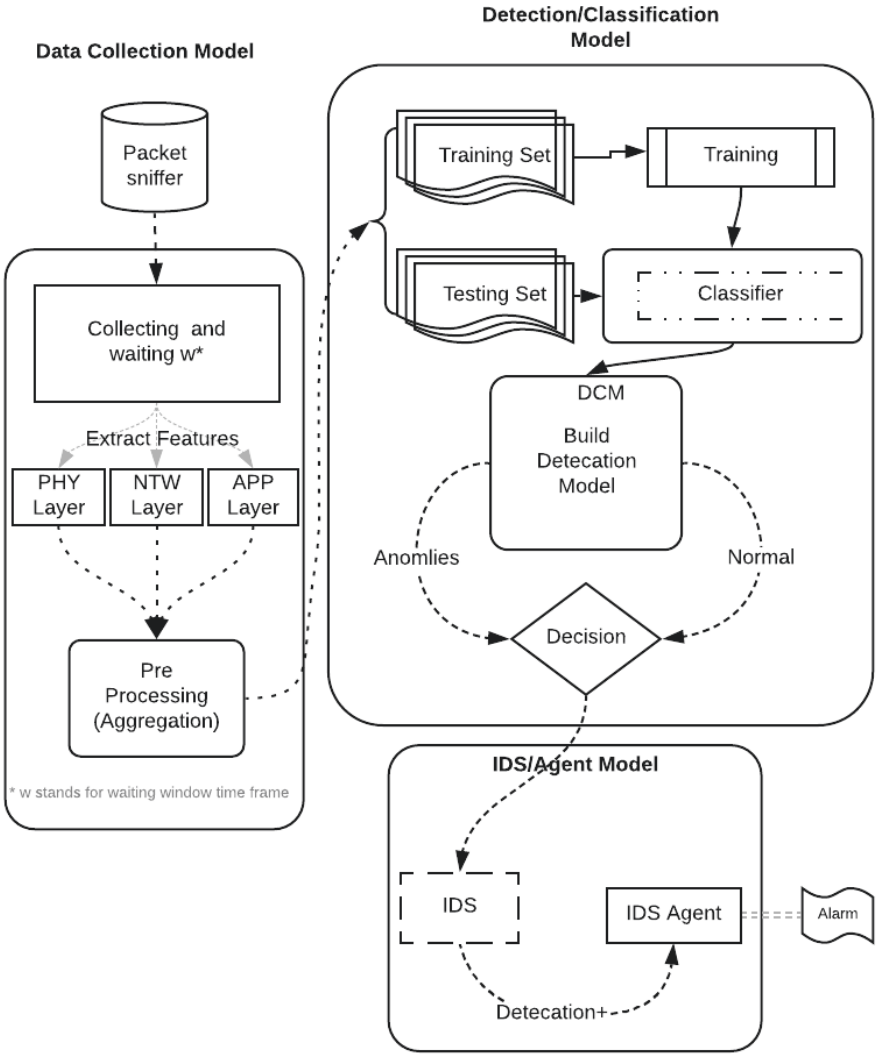


Figure 4.1 : IDD-IoT framework

in Figure 4.1. Due to the limitations explained in the previous chapter, where no dataset reflects the nature of the IoT network, the pre-learning phase is essential. All of the steps that take place before the real detection system is implemented in the network occur in this phase. This phase includes raw traffic data collection, feature extraction and selection, dataset creation and training, and testing the machine learning model. The post-learning phase comprises three steps, online data collection, the detection model, and the node agent model. These three steps are responsible for continuously monitoring the traffic data to search for any anomalies based on a set of predefined thresholds. A detailed overview of the proposed

framework is presented in Figure 4.1.

As presented in Figure 4.1, the framework compromises three modules. Figure 4.1 shows our proposed architecture which consists of three modules:

1. Data Collection Module(DCM)
2. Classification Module (CM)
3. Detection Modules (CM)

The data collection module (DCM) can be considered as a cross-phase module since it is proposed to be part of the two phases of pre-learning and post-learning. The detection module and the node agent module are both part of the post-learning phase and are responsible for attack detection and countermeasures. Furthermore, traffic monitoring, data classification and attacker isolation all occur at this phase. Each module of the framework is further explained in the following sections of this chapter. Comparing and selecting the best machine learning method is also explained since it is part of the pre-learning phase.

4.3 Overview for the Data Collection and Dataset Creation Framework

Before going into the process of data collection, a deep understanding of the type of network and protocols associated with the network must be explained. This thesis focuses on the IoT network that uses the IETF [10] standard for communication, which is the RPL protocol for routing and the 6LoWPAN protocol for IP adaptions. To further understand the DCM, the process of data collection is divided into the following two categories:

4.3.1 Data Collection Module

As described in Section 4.2 of this chapter, the data collection tool is a cross-phase tool that works in the pre- and post-learning phases. The first phase, the offline process, is to collect data to create an offline dataset to train and test the machine

learning algorithm. The second phase is where the data collection tool is integrated into the full IDD-IoT framework for online data collection and is then fed directly to the detection module, as shown in Section 4.3. The main goal of this module is to collect IoT communication data, either in a real network or in a simulated 6LoWPAN and RPL network, but our architecture is not limited to such protocols and can be generalized to any protocol. Before detecting any attack, we must acquire some data from the network. The acquisition process is divided into three categories: the physical layer features, the network layer features and the application layer features. Some existing datasets deal with the idea of DDoS attacks; however, these datasets need to be modified to comply with the IoT network requirements. For example, the KDD dataset has features of services that do not work in IoT networks, such as the SNMP protocol features. The KDD dataset has duplicates in records that can impact the result of the proposed machine-learning algorithm for detecting DDoS. Furthermore, the KDD dataset does not reflect the communication data of an IoT network. To meet this objective, this research develops a Python-based tool that gathers information from the IoT environment that is built using the Cooja simulator [77]. The main idea is to capture all traffic using a network sniffer and further analyze the traffic and store the packets as a database. The Python tool is used to extract the features that we want to use as a training dataset to train the machine-learning algorithms.

In the next step, the generated PCAP file from our IoT network is taken, and the relevant information for the dataset is extracted (feature selection).

4.4 Overview of the Machine Learning Methodology and Evaluation

There are hundreds of machine learning algorithms from which we can choose for comparison. However, since the IoT environment is resource-constrained, only three algorithms are chosen based on two factors: performance and accuracy in a limited resource network environment. The following three algorithms have been widely used in resource-constrained networks for different applications. However,

none of them have been compared for usage in IoT environments; specifically, DDoS attack detection applications. The following subsections contain a brief explanation of each algorithm.

4.4.1 Artificial Neural Networks

The ANN machine learning method is a widely used method for different applications, including security. In a nutshell, the ANN method is inspired by the human brain neural network and simulates how the brain constructs a memory connection to form a neural network. The ANN uses probability to build weighted connections between the input and the output. One of the popular types of ANN is a special neural network called the multi-layer perception neural network, which is used in this thesis. This is further discussed in Chapter 7 of this thesis.

4.4.2 Support Vector Machine

An SVM is a very popular machine learning method that has proven its reliability and accuracy for different applications; one of these many applications is anomaly detection in security-related research. Examples in [87] and [88] show how SVM can be used for security-related research.

4.4.3 Decision Tree (Random Forest)

The decision tree is a machine-learning algorithm that is very popular among the machine-learning community, specifically in the area of data mining. [89] states that this is one of the most widely used machines learning algorithms in most applications. It applies the same concept used in the ID3 algorithm, where it makes decisions based on the concept of information entropy [90].

4.4.4 Detection Methods

Figure 4.2 below shows the proposed detection module. This module process will be repeated for each of the proposed methods. The outcome will be analysed later and the best performer and most accurate algorithm will be chosen based on the metrics discussed as follows:

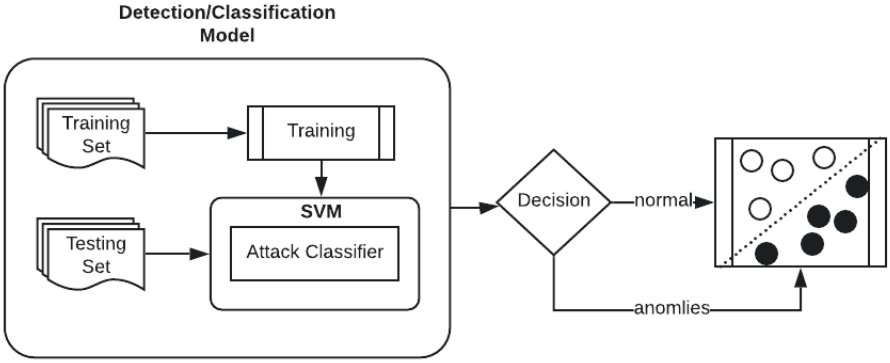


Figure 4.2 : Detection method

This research also develops a Python system based on the SciPy library and other libraries, as outlined in the following steps.

- **Step 1:** Develop a parser that can parse the collected data into the machine-learning algorithms, looking for the algorithm that produces the best result in terms of performance and detection.
- **Step 2:** Calculate the metrics for each algorithm based on the selected attributes detailed in Section 4.4.5 to calculate accuracy and to determine the false positives and false negatives.
- **Step 3**: Build a detection module based on the best result produced in Step 2. This module should detect the anomalies in the network traffic since the module will be embedded in the IDS solution that will be implemented later in the network.
- **Step 4:** Save the detection module generated from the best algorithm in Step 3 to a file that can be embedded later in the IDS.

4.4.5 Machine Learning Validation

One of the basic metrics used to validate the machine learning algorithm's performance is called the confusion matrix. In its basic form, the confusion matrix is represented using a table with all of the predicted values compared to the ground-truth values. The rows in the confusion matrix represent the predicted values for

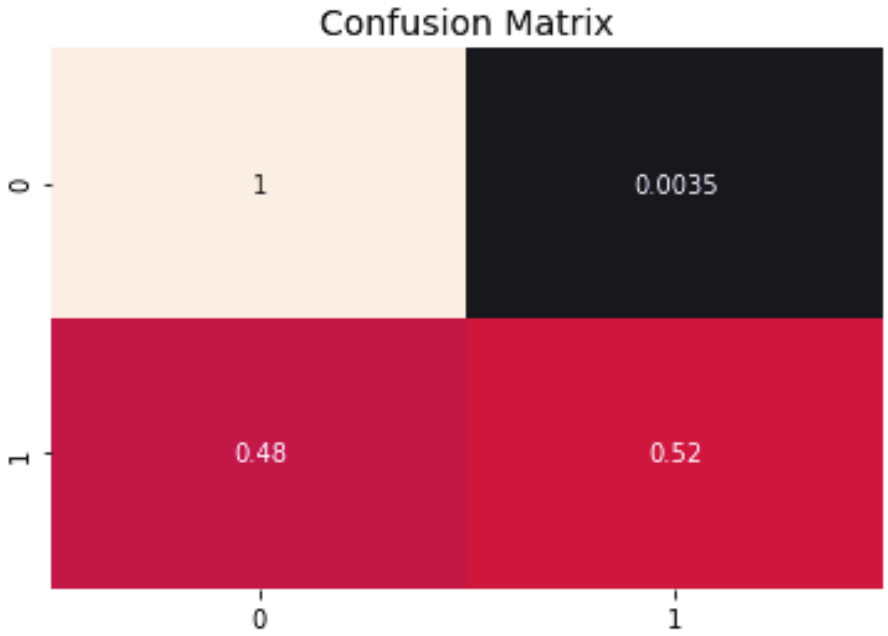


Figure 4.3 : Example of a confusion matrix

each class, whereas the columns represent the ground-truth values. Figure 4.3 shows an example of a confusion matrix. The confusion matrix has the following parameters:

- True Positive (TP): This represents the samples that are correctly flagged as normal.
- False Positive (FP): This represents the samples that are incorrectly flagged as a normal attack.
- True Negative (TN): This represents the samples that are correctly flagged as an attack.
- False Negative (FN): This represents the samples that are incorrectly flagged as normal but are actual attacks.

Extracting the confusion matrix will help us to calculate the following metrics:

Accuracy: The accuracy problem is the most straightforward kind of metrics for evaluating a machine learning algorithm's performance. Its basic form calculates the

number of correct predictions divided by the total number of the whole predictions. It is a useful metric for gaining the overall score of the algorithm used.

$$Accuracy = \frac{NumberofCorrectPredictions}{TotalNumberofPredictions}$$

Precision: In many cases, accuracy metrics are not sufficient to evaluate how well a machine learning model is performing. One of these cases is when the dataset has more data points in one class than the other, leading to an imbalanced classification model. Furthermore, and since we are dealing with a majority of normal traffic in our scenarios and the attackers are the anomalies, the normal instances are far more in number than the malicious instances. Therefore, the need for more precise metrics is a necessity to ensure a robust machine learning model. This is where precision calculates the accuracy of each class representation using the parameters from the confusion matrix and is represented as follows:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall: Also sometimes called detection rate, the recall is the ratio between the number of attacks detected by the system and the total number of attacks that are present in the dataset. It is represented as follows:

$$Recall = \frac{TruePositive}{Truepositive + FalseNegative}$$

F1 Score: This metric depends on the type of application; in some cases, more priority should be given to precision and sometimes to recall. However, in some applications, both metrics have equal importance, therefore combining them forms the F1 metric, which is represented as follows:

$$F1 = 2 * \frac{Precision + Recall}{Precision + Recall}$$

The F1 score allows us to identify how precise the classifier is and how robust it

is when it comes to ignore undetected instances. The generalized version of the F1 Score can be represented as follows:

$$F_\beta = (1 + \beta^2) * \frac{Precision * Recall}{\beta^2 * Precision + Recall}$$

A further explanation of these matrices with their applications within the scope of this thesis is discussed in Chapter 7.

4.5 Overview of the ML IDS Implementation in the IoT Network

The solution proposed in this research consists of different modules: the nodes, the detection module and the agent in each node. Figure 4.4 shows the framework for the post-learning phase which is the actual implementation in Cooja.

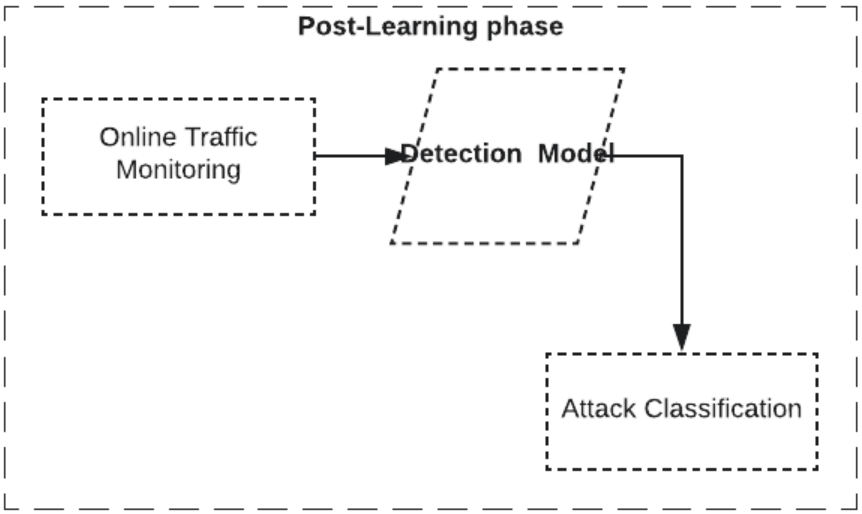


Figure 4.4 : Post learning phase

After evaluating the best method for countering DDoS attacks in the IoT that produces the best result with the fewest false positives and false negatives where the entire detection module is built from the previous objective, the best result machine learning algorithm is taken and implemented in the proposed IoT network. Figure

4.5 shows the detection module with different components and their placement in the network environment.

4.5.1 Proposed Network Diagram:

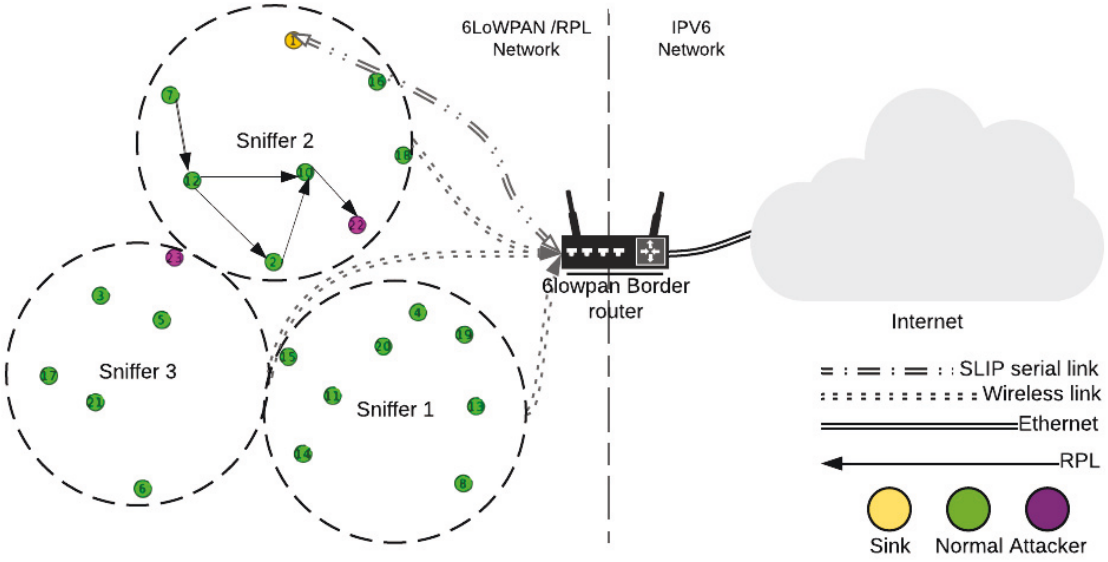


Figure 4.5 : Proposed network diagram

The network is designed, taking into consideration natural influencers like the humidity and the wind, which may affect the network's performance in general and radio transmission. Hence, a node is designed that transmits data at a constant rate to add noise to the environment. In this scenario, it is assumed that the 6BR router and the sink node have more power than regular nodes. Since Cooja allows the network to be bridged to another implemented 6BR router, the 6BR router is easily installed in another Linux-based machine and acts as the router.

4.5.2 IDS Agent Module:

- **Step 1:** The IDS agent monitors the traffic for any suspected attacks. The suspected attacks are measured using the increase in DIO and consistent rank changes in the RPL protocol. If the DIO messages exceed certain thresholds combined with two consecutive changes in the node's rank, an acknowledgment is sent to the IDS.

- **Step 2:** After receiving the acknowledgment, the IDS further investigate the alert and, using the machine-learning algorithm, the IDS decides whether the data are normal traffic or anomalies.

- **Step 3:** If an attack is detected, the IDS sends an alert to the IDS agent to isolate the node from the network and blacklist it.

- **Step 4:** The IDS agent broadcasts an alert message to all of the nodes in the network to isolate the attacker from the network and blacklist it.

4.5.3 IDS Evaluation Overview

In order to further evaluate and validate the approach proposed in this study, two scenarios are developed. These scenarios are as follows:

Scenario 1: The IoT-DDoS dataset is evaluated by building a machine learning model for three kinds of machine learning methods to evaluate the performance for each method.

Scenario 2: The best performing machine learning method is implemented in a simulated environment incorporating the data collection model and the detection model utilized in this thesis. Furthermore, the system is evaluated for both scenarios with and without the developed IDS and it is benchmarked with various attacks introduced.

The metrics used for evaluation are the same as those described in Section 4.4.5.

4.6 Conclusion

In this chapter, a brief description of the proposed machine learning framework for DDoS detection in IoT is discussed. All the different artifacts that help address the objective of this thesis are identified. The chapter starts by giving a general framework for the solution, including a definition of each component of the framework. This was followed by a brief explanation of the data collection models and the dataset generation process.

Chapter 5

A Machine Learning Architecture for Detecting Denial-of-Service Attacks in IoT

5.1 Introduction

The Internet of Things is part of everyday life, where millions of devices are connected to the Internet to collect and share data. Although IoT devices are evolving quickly in the consumer market where smart devices and sensors are becoming one of the main components of many households, IoT sensors and actuators are also heavily used in industry where thousands of devices are used to collect and share data for different purposes. With the rapid development of the Internet of Things in different areas, it is difficult to secure the overall availability of the network due to its heterogeneous nature. There are many types of vulnerabilities in the IoT that can be mitigated with further research; however, in this study, we concentrate on the distributed denial-of-service (DDoS) attack on IoT. In this study, we propose a machine learning architecture to detect DDoS attacks in IoT networks. The architecture collects IoT network traffic and analyzes the traffic passing through to the machine learning model for attack detection. We propose the use of real-time data collection tools to monitor the network dynamically.

5.2 Framework

In developing any kind of solution for IoT scenarios, constraints, and limitations have to be considered. Developing any machine learning solution requires the acquisition of data related to the area under research. Up to this time, to the best of our knowledge, there is no predefined dataset designed for IoT networks specifically. The problem with previous datasets, like the KDD dataset, is that it was designed and collected for a different set of protocols that are not used for IoT networks. There-

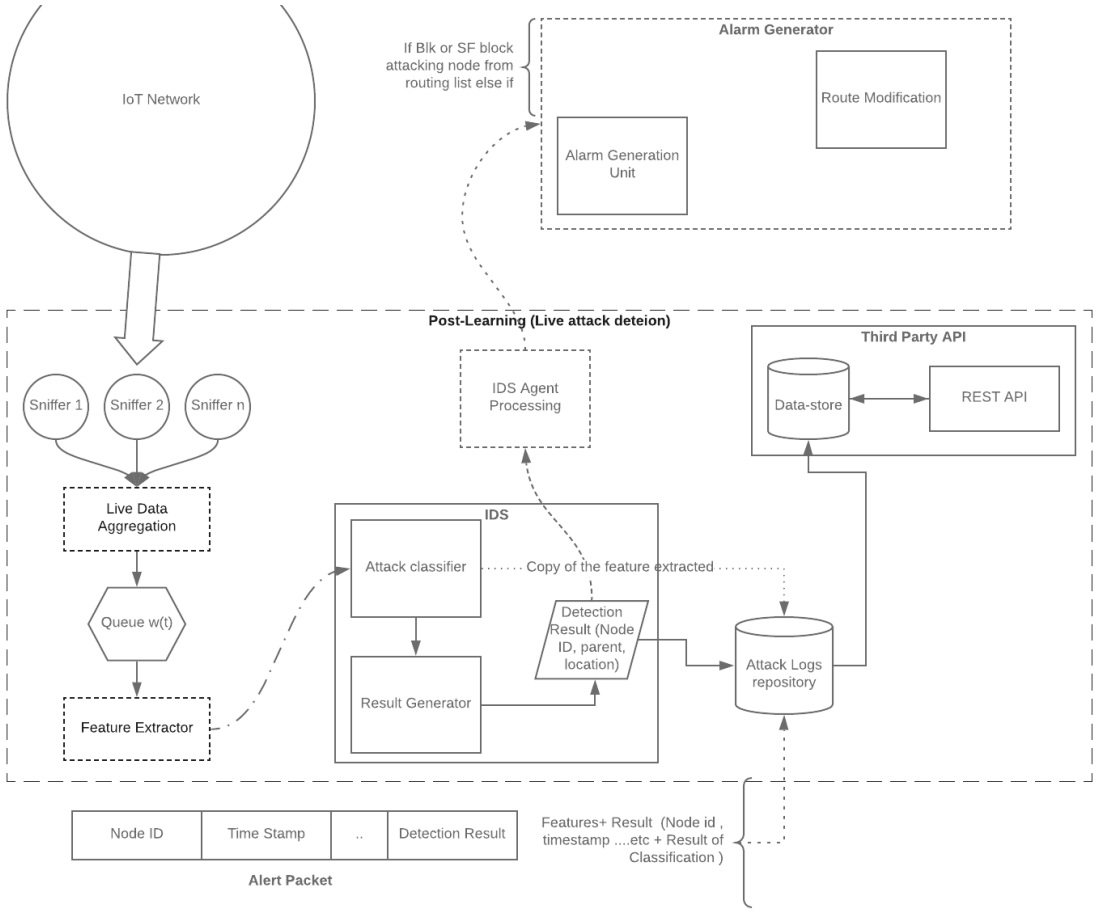


Figure 5.1 : Intelligent DDoS detection framework (IDD-IoT)

fore, in designing our architecture,we propose to design and collect a new dataset to detect attacks and encounters in the IoT networks. This framework focuses on IoT-related protocols such as the 6LoWPAN and RPL protocols. To design an effective machine learning framework, we divide the framework working strategies into two phases, as we explained in Chapter 4. Figure 5.1 shows our proposed architecture consisting of three modules:

1. Data Collection Model (DAM)
2. Classification Model (CM)
3. Detection Model (DM).

Figure 5.2 shows the two phases and the associated task in each phase. As shown,

the data collection module is shared across both phases since the data collection, and traffic monitoring take place in both phases. At the pre-learning phase, the data collection module is used to collect offline data for training and testing three machine learning algorithms. In the post-learning phase, the DCM is used for deep packet inspection and monitoring based on the machine learning model built in the previous phase.

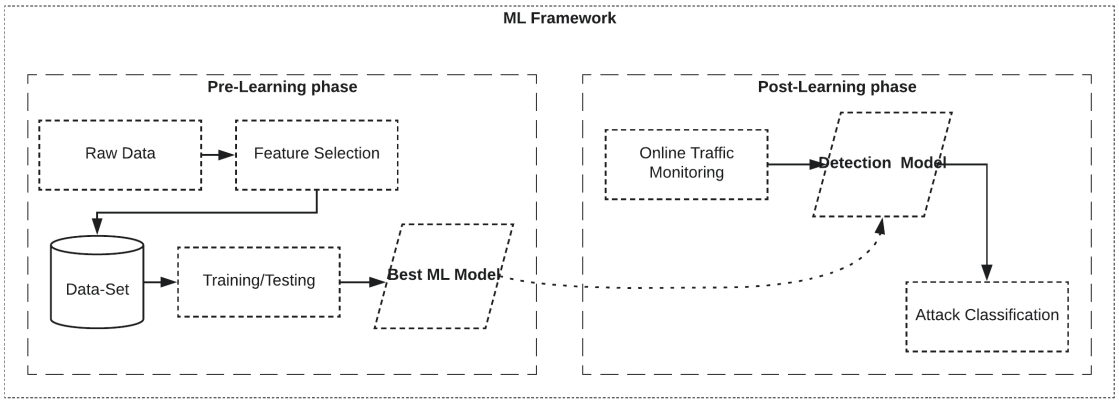


Figure 5.2 : Framework phases

The following section explores each of the three modules in more detail.

5.2.1 Data Collection Module

The main goal of this module is to collect IoT communication data, either in a real network or the simulated 6LoWPAN and RPL network, but our architecture is not limited to such protocols and can be generalized to any protocol. Before detecting any attack, we must acquire some data from the network. The acquisition process is divided into three categories: the physical layer features, the network layer features, and application layer features.

Physical Layer Features:

Where the DCM extracts physical layer related features such as the received and transmitted signals at the MAC layer, this information is related to physical layer jamming attacks, which usually interfere with the transmitted signals.

Table 5.1 : Data collection features

Feature	Description
Physical Layer Features	
Received signal DBM	Mean value of the received signal at the MAC layer
Transmission signal DBM	Mean value of the transmitted signal at the MAC layer
RSSI noise	Mean value of the noise recorded using RSSI
Beacon interval	Mean value of the beacon interval
Network Layer Features	
LQI	Link quality indicator
ETX	Mean value of the expected transmission count
Number of DIS messages	Number of DIS messages
Number of DIO messages	Number of the RPL DIO messages
RPL rank	Number of rank changes over time.
Number of neighbors	Number of neighbours
Application Layer Features	
Temperature	Mean value of the temperature
Humidity	Mean value of the humidity
Power level	Mean Value of the energy over time
Consumed power	Mean value of consumed node power
Remaining power	Mean value of the remaining node power
Node ID	Node ID

Network Layer Features:

The collection of network layer features is crucial for our IDS since the features extracted are closely associated with many famous attacks, such as deceased rank attack and blackhole attacks [16]. In this category, features such as the number of DIS and DIO messages are extracted, and some other features which are related to the RPL protocol. Data transmitted through any network follow certain protocols. In IoT, there are certain protocols available at every layer, and in this study, we focus on the RPL protocol on top of the 6LoWPAN protocol. This can be expanded to include extra features from other protocols.

Application Layer Features:

At the application layer, our module collects application-specific information such as humidity, temperature, and node power level. The application-related fea-

tures can be extracted by programming the nodes to calculate the power consumption and other related features. We chose to collect power consumption at the application layer because we can program the node to calculate the power level and send the result within the application-related information in the data frame payload.

5.2.2 Feature Selection

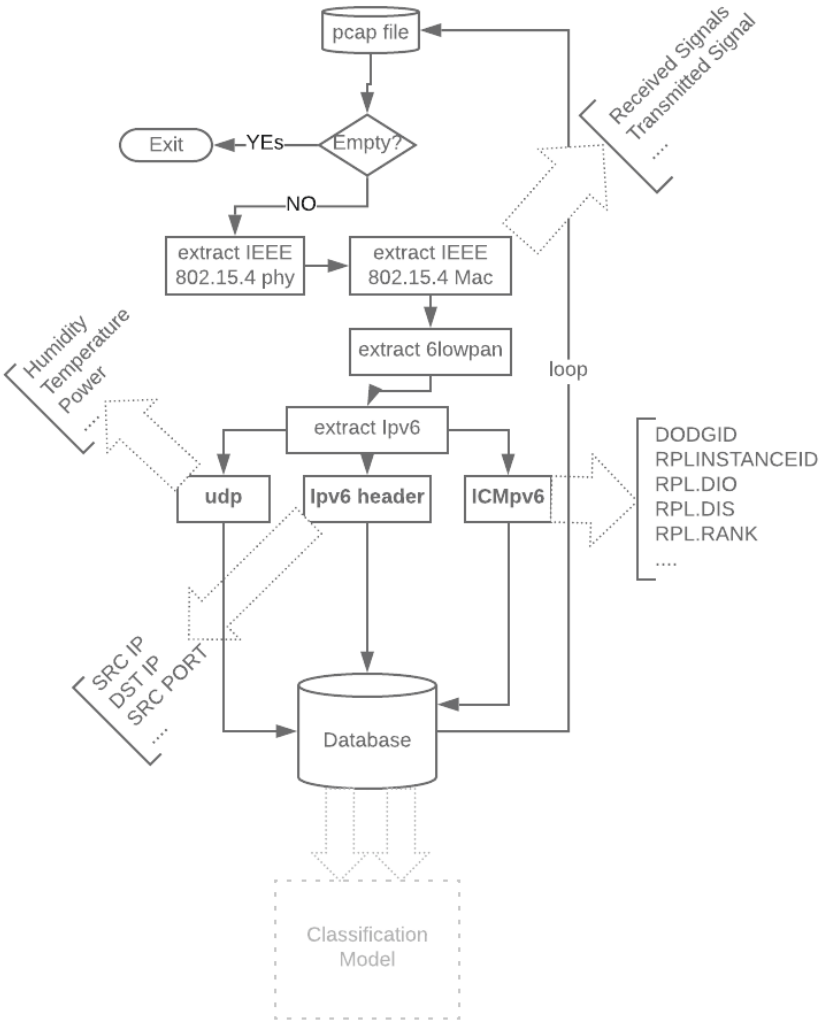


Figure 5.3 : Data collection module

To extract the network-related features, we design a specific tool that follows the 6LoWPAN and RPL protocol structure. Figure 5.3 shows the process involved in extracting the specific features from the Pcap file captured from the IoT network. To acquire raw IoT network traffic, we propose the use of the sensniff [91] sensor to

cover the entire network. The sniffed data are then fed to our DCM tool to extract the features shown in Table 5.1. To extract the 6LoWPAN and RPL features, we design our tool to exploit both protocol structures. Figure 5.3 details the process of extracting packets transmitted in 6LoWPAN networks. At each layer, a set of features are extracted, as previously mentioned. Furthermore, before extracting the features, a time window must be specified to aggregate the data into records. This timeframe window will be used later to obtain the deviation or average for each node based on that time window. Depending on the IoT network application, the time window can vary accordingly. Designing a dataset for IoT electricity metering will have a different effect if the same dataset designed with the same time window is used for parking sensors. Since each application has a different level of data generated during communication, determining the time window depends on the type of IoT application used and the type of protocols implemented. The primary goal of this dataset is to train and test the machine learning algorithm and generate a detection model, which is explained in the next section.

5.2.3 Data Classification Module

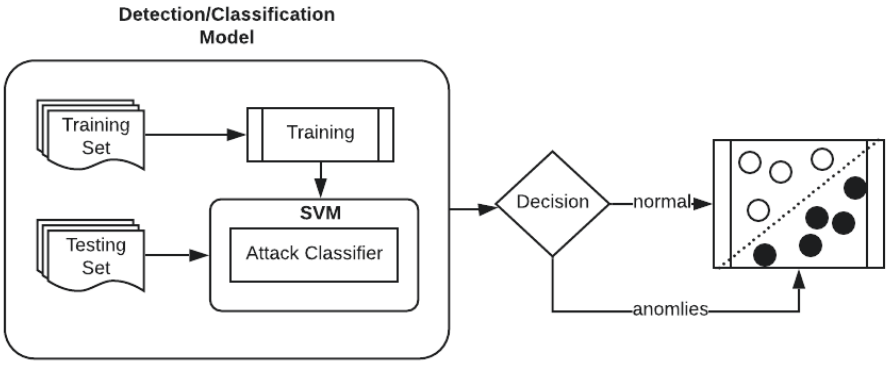


Figure 5.4 : Example of training ML model

The dataset generated from the DCM will be used for training and testing our machine learning algorithm. At this level, we experiment with different machine learning methods, and the one with the best results in terms of accuracy and performance is selected. The quality of the results produced at this stage depends heavily

on how the data was collected in the previous stage. Figure 5.4 shows an example of how to train and test SVM for attack detection. As previously discussed , building a machine learning model consists of two stages, each of which is explained further in the next section.

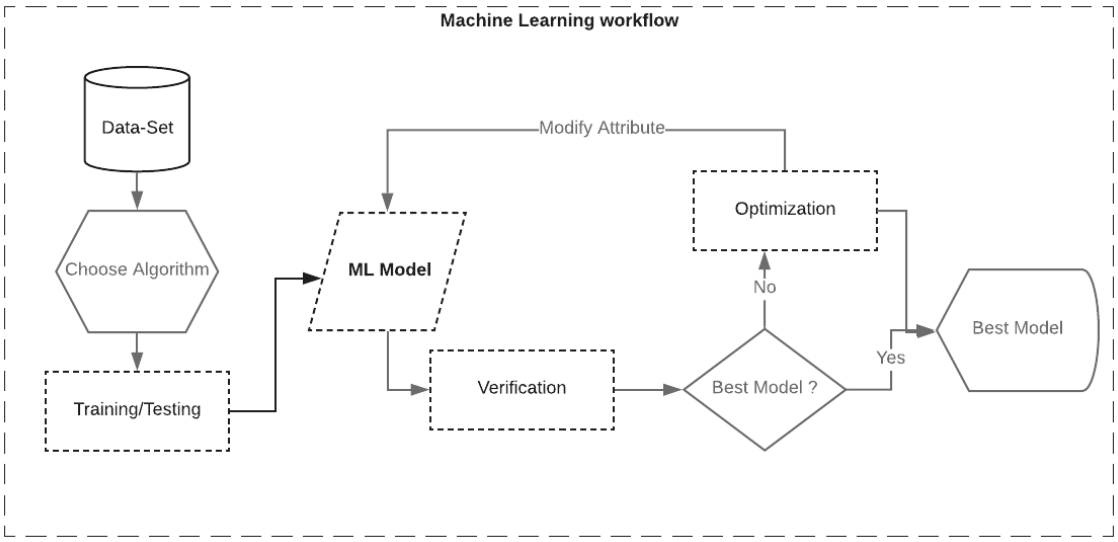


Figure 5.5 : Pre-learning workflow

Pre-learning Phase

The pre-learning phase is where the machine learning model is trained and tested based on the collected data. This thesis focuses on three commonly used machine learning methods and tests each one to identify the best performing method for the scope of this research. Figure 5.5 shows the workflow to select the best model for each method chosen. The part of the pre-learning phase which was discussed in the previous section is the data collection and feature selection part. The process of selecting the best machine learning method can be explained in the following steps:

1. **Choosing Algorithm:** before training the model, machine learning must be chosen. As previously stated, this thesis focuses on three types of machine learning methods (random forest decision tree, SVM, neural network)

2. **Training/Testing:** this is the learning phase for the machine learning model,

where the data collected as a dataset is fed to the chosen algorithm to produce a machine learning model.

3. **Verification :** in this step, the model is verified using a set of attributes and scores, which is explained in detail in the next section.

4. **Optimization :** in this step, the verified model, is iterated into multiple iterations until the best result for each machine learning algorithm is obtained.

At the end of this iteration, three optimized machine learning models are generated. Based on the results in the verification phase, the best model will be chosen and deployed into our IDD-IoT system for attack detection.

To detect attacker nodes, we developed the proposed algorithm, which is discussed in the next section. The validation process is explained in the following.

Post-Learning Phase

The post-processing module is responsible for handling real-time data and actions. This part of the detection module is divided into three steps.

1. **Traffic aggregator:** The main purpose of this step is to aggregate data from multiple sniffers installed in the network. Supporting multiple sensors in the network is essential to ensure network scalability and attack detection coverage. Therefore the aggregation process must ensure the data fed to the feature extraction process is not duplicated. To ensure this process occurs smoothly, this section of the thesis proposes Formula 5.1 to aggregate data in real-time and ensure clean data formation. The formula is described below, assuming the number of sniffer sensors is S= (S1,s2,s3 ... sn), and the number of nodes is N = (n1,n2.n3...nm) where m is the last node in the network. The process of ensuring that the sniffer node only aggregates non-duplicated data is to compare the received packet signature at the T time and ensure they are not the same. In this work, the signature is defined as the variable Ts, which consists of the timestamp and node ID. If the signature of the packet equals

any packets received from any other sniffer, one of the packets will be ignored, and only one version of the packet is processed in the queue. Otherwise, no additional process is needed, and the packets are forwarded to the next set. This process ensures no data duplication occurs in real-time. This unit consist of three handlers explained as follows:

- Sniffer Handler(Sniffer[]): this handler takes the sniffer arrays as input and processes each packet coming from each sniffer. The output from this handler is passed to the aggregation handler. The output is a multi-dimensional array of sniffers and packets associated with each sniffer.
- Aggregation Handler(Sniffers[node][packet]): The aggregation handler iterates through all of the sniffers and their packets to find any packet signature similarities and report them. After iteration, only one array of node packets is sent to the queue.

$$g(R_x, R_y, R_z) = \begin{cases} 0 & \text{if } R_x \in [X_1, X_{\text{end}}] \wedge R_y = f(R_x) \wedge R_z < C \\ 1 & \text{otherwise} \end{cases} \tag{5.1}$$

2. **Queue Unit:** This unit is responsible for queuing packets to allow exact feature extraction based on a time window wt. The time windows are determined based on the type of IoT application used. Allowing this flexibility in choosing the time window in the framework is essential to ensure a scalable solution. Algorithm 2 describes the process executed in this unit.

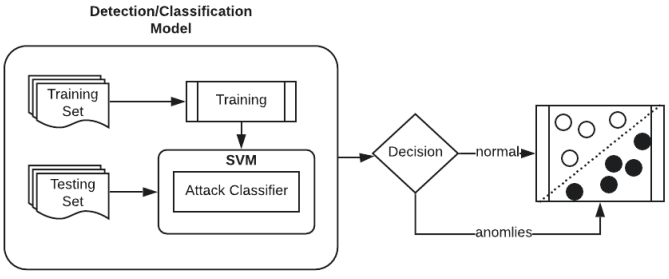


Figure 5.6 : Data detection model

Algorithm 1: Queuing process algorithms

```
Result: S
Queue = array of packets;
packets = incoming pacekts; while packets != 0 do
    tmp = pacekt[0].timestamp - packet[1].timestamp;
    if condition then
        instructions1;
        instructions2;
    else
        instructions3;
    end
end
```

It is worth mentioning here that the *wt* time windows used in the pre-learning process should be the same to ensure precise, unbiased data collection.

3. **Feature Extraction:** the same feature extraction process described in Section 5.2 is applied here, but instead of being offline, the process is online on real network scenarios.

4. **Attack classification:** this is the step where the chosen machine learning classifies the traffic anomalies based on the testing and training process described in Section 5.2. As previously discussed, the best machine learning model will be embedded into the IDD-IoT framework to ensure the best result. More on this part is discussed in chapters 7 and 8. Figure 5.6 shows an example of the anomaly detection using the one-class SVM.

5. **Result Generator Unit:** this unit is responsible for generating the result of the classification and creates a special kind of packet on top of the UDP protocol and reports to the IDS agent where further processing is handheld by the IDS agent. It is worth mentioning that the generated packet has a minimal size footprint to ensure no network traffic is introduced. This packet is sent using a payload in a UDP packet, and its size is less than one byte. The packet contains the following parameters(node-id, time, parents, rank, and detection result). The detection result can be either 0 or 1. If the result is 0, it indicates

no attack was detected, and no further packet will be sent to the IDS agent. This result is then stored on the local log repository. Storing the result on a local repository is crucial for future analysis. Otherwise, if the detection result is 1, the detection packet is sent to the IDS Agent, and a copy of the result is stored on the local repository as well.

6. **Local Repository (LR):** This local database stores all of the features' extracted records alongside the result from the attack classifier. The main reason for creating such a repository is to build a knowledge base for future analysis and model improvement. Moreover, it is the data store from where all of the third-party units retrieve the data. Using this repository allows a third-party top-level application to visualize in real-time how the network is performing. Figure 5.7 shows all of the information stored in the repository. Moreover, to handle all of the retrieval and posting processes, two handlers are created with the database which are explained as follows:

    - **GetData() Handler :** responsible for extracting the data from the datastore and posting it to the required application using the REST API.
    - **PostData() Handler :** responsible for inputting the data from the sources to the database. This handler is usually triggered by the feature extraction unit and the result generator unit, to store the information to the database.

7. **API Unit:** this unit works as the interaction point between the end-user and the local datastore. In this thesis, the three-tier REST API model is used. The basic design of any REST framework includes four operations, which are commonly referred to as the CRUD operations and can be described as follows:

    - POST: Create data
    - GET: Retrieve Data
    - PUT: Modify Data
    - DELETE: Destroy Data

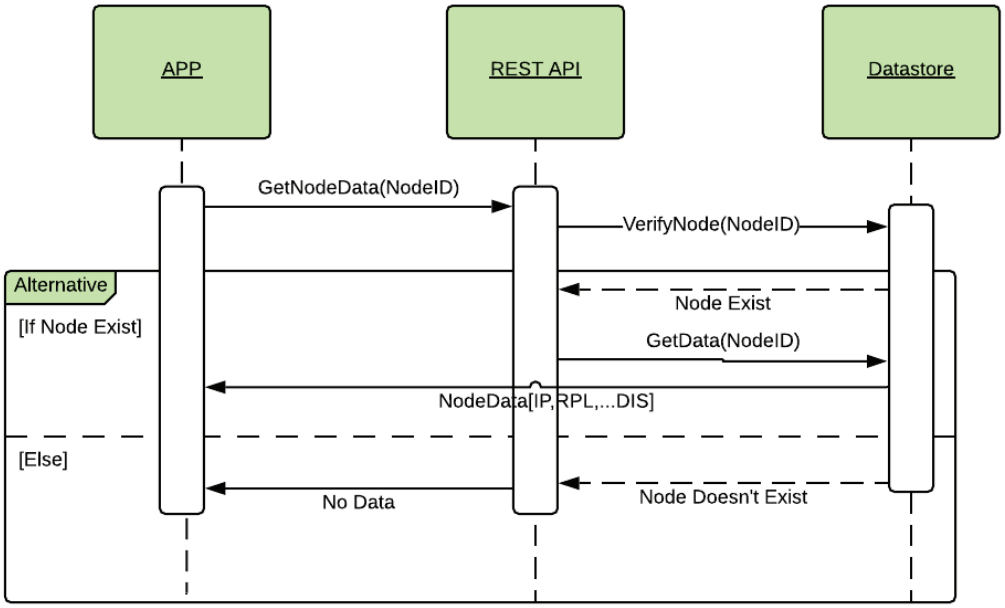


Figure 5.7 : Workflow of the GET operation

The API has two components, the readData(NodeID) and the update(NodeID), where the readData() function is used to fetch the data from the data store using the MySQL interaction and send the data to the third party front end. The Update(NodeID) is used to update any misinformation manually. This function is only used by administrators of the network and does not affect the actual network itself to protect it from unauthorized misbehavior. To ensure a bottleneck-free API, this unit was designed using the three-tier REST API architecture. Figure 5.7 shows the full workflow of the three-tier API and each interaction level, and it also shows the detailed workflow of how the GET operation is handled. As can be seen, this unit is divided into three tiers and is described as follows:

- **Endpoint client or third-party application:** this part is where the client application sends a request to the REST API to retrieve certain data from the data store.
- **REST API Framework**: this part works as the middleware between the data store and the client front-end.
- **Data Store:** this is where all of the node information and the detection

state is stored. The data store object has its handler to handle incoming requests, which is designed to make sure the REST API has no direct access to the data store, which adds an extra level of abstraction to the end-user.

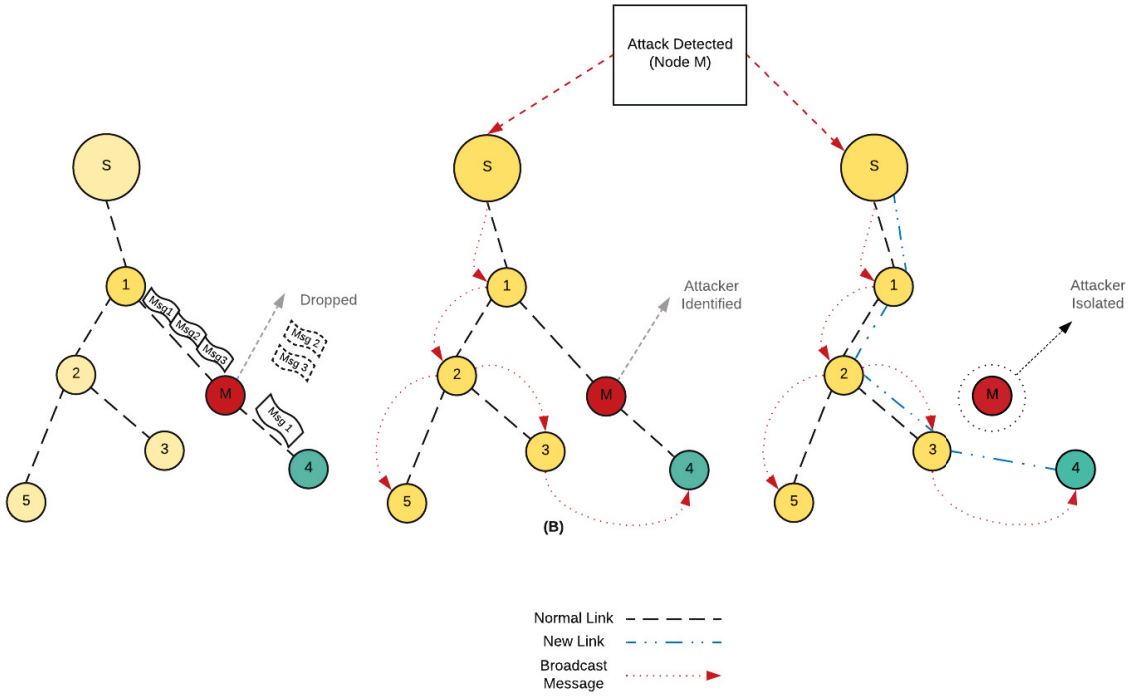


Figure 5.8 : Example IoT network

5.2.4 IDS Agent Module

This module works as the connection point between the local network and the IDD-IoT. It is built on top of the sink node of the network since all nodes are connected to the sink node either directly or through a multi-hop. The main idea of this module is to broadcast an alarm message to all nodes of the network containing the attacker ID and the path to the attacker. This will allow another unaffected node to add the attacker node to the blacklist and avoid any communication with the malicious node. Furthermore, the IDS agent modifies the victim node's route and creates a new alternative path to the sink node. Then, the IDS agent initiates network topology reform to isolate the malicious node by establishing a new route

to the sink from the victim node. All nodes will blacklist the malicious node, and all traffic from it is ignored and dropped. Figure 5.8 shows a simplified diagram of the whole process explained here, where S represents the sink node and M is the attacker node. To achieve such a result, the IDS agent implements two main functions :

- **AlertBroadcast(AttackerNodeID):** As discussed above, this method's main responsibility is to broadcast alert messages containing the AttackerNodeID to all of the nodes within the network. This can be achieved by utilizing the DIO message available on top of the RPL protocol.

- **RouteModify():** This method has the responsibility to reroute the traffic which is coming to and from the victim node using the nearest neighbor with the highest rank within the DODAG tree.

5.3 Conclusion

Detecting DDoS attacks in IoT networks is a challenging issue, especially in industrial IoT applications. Due to the consequences associated with such an attack which affects the availability of the network and its services, these cause a significant loss in money and information. It is, therefore, crucial for IoT providers and operators to mitigate such attacks in a fast and efficient manner. The IDD-IoT framework assists providers and administrators mitigate DDoS attacks without compromising the limitations associated with limited resource devices. If an attack is detected, the framework assists providers in detecting DDoS attacks. In this framework, the power of machine learning is utilized to detect any behavioral change in the network by continuously monitoring the traffic based on a set of properly selected parameters.

In the next chapter, the data collection model is implemented to collect the proposed dataset. Furthermore, the structure of the dataset and its related features are discussed.

Chapter 6

Real-time Dataset Generation Framework for Intrusion Detection Systems in IoT

6.1 Introduction

Intrusion detection systems (IDS) are not a new concept; rather, they have existed since 1970 [92]. The main component of any proper IDS is a very well structured and gathered dataset; an accessible dataset widely used in many security studies is the KDDcup-99, which was considered the golden standard for IDS evaluation for a long time [93]. However, the KDDcup-99 suffered from many flaws, which were criticized repeatedly on many occasions, such as in [94] [95] [96] which indicates that it is not applicable for today IDS's and security research. Furthermore, the scope of this research is IoT-based networks; and as we discussed in Chapter 2, the literature lacks dedicated IoT based IDS datasets. Therefore, this chapter is dedicated to building a new set of tools to evaluate and generate a new IoT based dataset to be used for the machine learning IDS framework.

In this chapter, we propose a new framework for generating an IoT dataset for IDS evaluation, based on the identified limitations of the existing datasets in Chapter 2.

The rest of the chapter is organized as follows: Section 6.2 explores a new pre-evaluation tool to be used for exporting related data for evaluation purposes. In section 6.3, the proposed framework is explained, including feature engineering and selection. The datasets are analysed in section 6.4. In section 6.5, we compare the results of the produced datasets and compare them with several existing datasets. Finally, conclusions are drawn in section 6.6.

Note: Some parts of this chapter have already been published in the FGCS [97]

journal and the NBiS [97] conference.

6.2 Data Exportation Framework

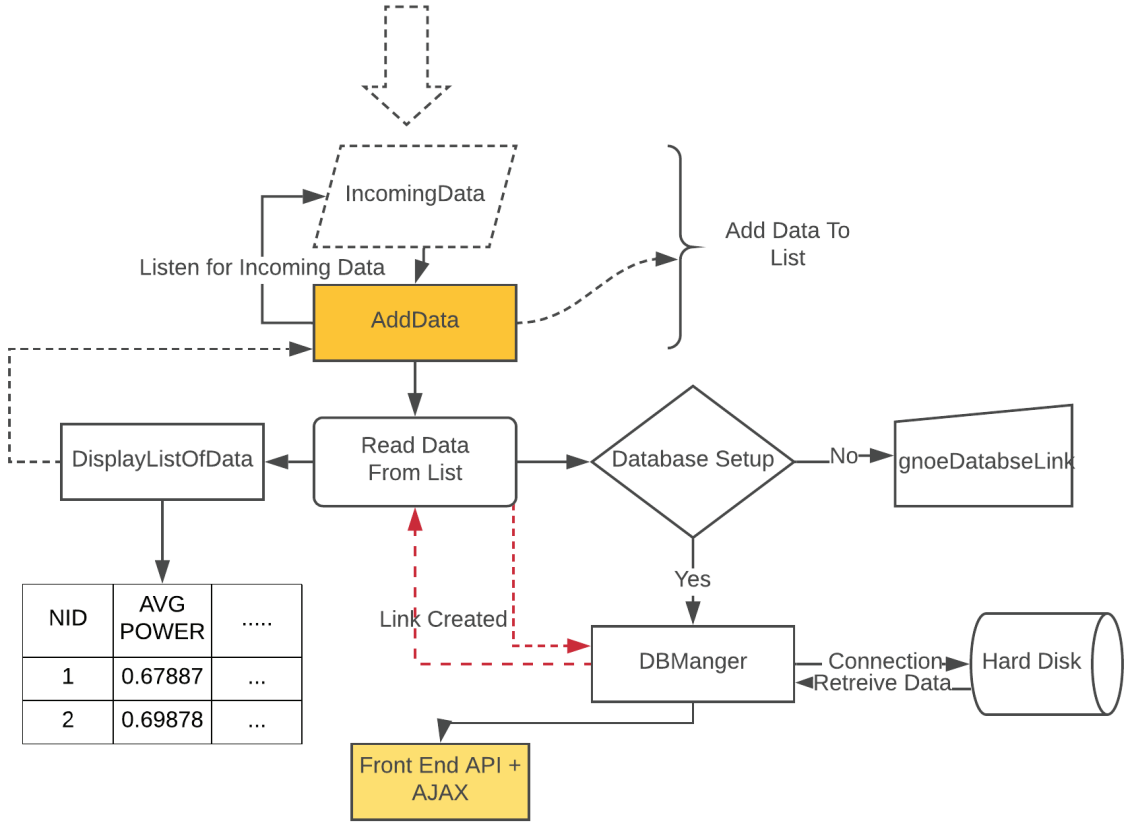


Figure 6.1 : DEF tool

Before we explain the framework, we develop a data exportation and visualization tool to evaluate the IoT traffic on the Cooja simulation platform. This tool provides insight into how the RPL and 6lowpan traffic is affected when different attacks are introduced, allowing us to focus on selecting the related feature. The Cooja simulator was developed to emulate IoT-based operating systems. This is a Java-based simulator that uses the Java Native Interface (JNI) to interface between Java and the emulated OS, which is in our example of the Contiki OS. A full sensor emulation is achieved through the use of MSPSim[98].However, Cooja lacks a tool for exporting the collected data for in-depth analysis. Acquiring sensor data and network-related data in a simple textual format will allow the researcher to inte-

grate it into other platforms and analyze them as needed. DEF is designed to be integrated as an extension for the available Collect View plugin in Cooja. DEF simplifies the process by gathering and exporting the collected data into a CSV format file or a MySQL-connected database. Such a tool allows researchers and developers to focus more on their tasks than to figure out how to export data to the correct format. To develop the DEF platform, we reverse-engineered the Collect View model, profoundly modifying the existing code to run the DEF tool during run time. The DEF framework consists of four main modules: the Data Handler Module (DHM), Data Visualization Module (DVM), Database Management Module (DMM), and API Module (APIM).

6.2.1 Data Handler Module (DHM)

Figure 6.1 shows the components of the DEF tool. The DHM module extracts the collected data for each node and stores them as an ArrayList to be embedded later in the table. To avoid blockage in the main thread, we separated the data-gathering process into two threads, one for GUI and the other for data handling and representation. The main task of this module is to handle, store, and update the incoming sensor data. All the nodes in the network and their information are

Algorithm 2: DEF handle incoming sensor data

Input: A set of Sensor Nodes Data $A = \{a_1, a_2, \ldots, a_n\}$
Output: Aggregated Sensors Data
$SensorDataList \leftarrow a_1$
for $i < a_n$ ***to*** n **do**
    if $a_i! = null$ **then**
        add a_i to the $SensorDataList$
        Update $SensorDataList$
    else
        Printout "Data is null"
return $SensorDataList$

stored and updated in an ArrayList, which is later embedded into the data DVM. To address all of this, we designed Algorithm 2 which retrieves a list of all nodes in the network and extracts the relevant sensor data as a data aggregator class. If

the retrieved node does not have any data, the process of adding sensor data to the sensor list is ignored. The received sensor list is passed in real-time to the DVM module.

6.2.2 Data Visualization Module (DVM)

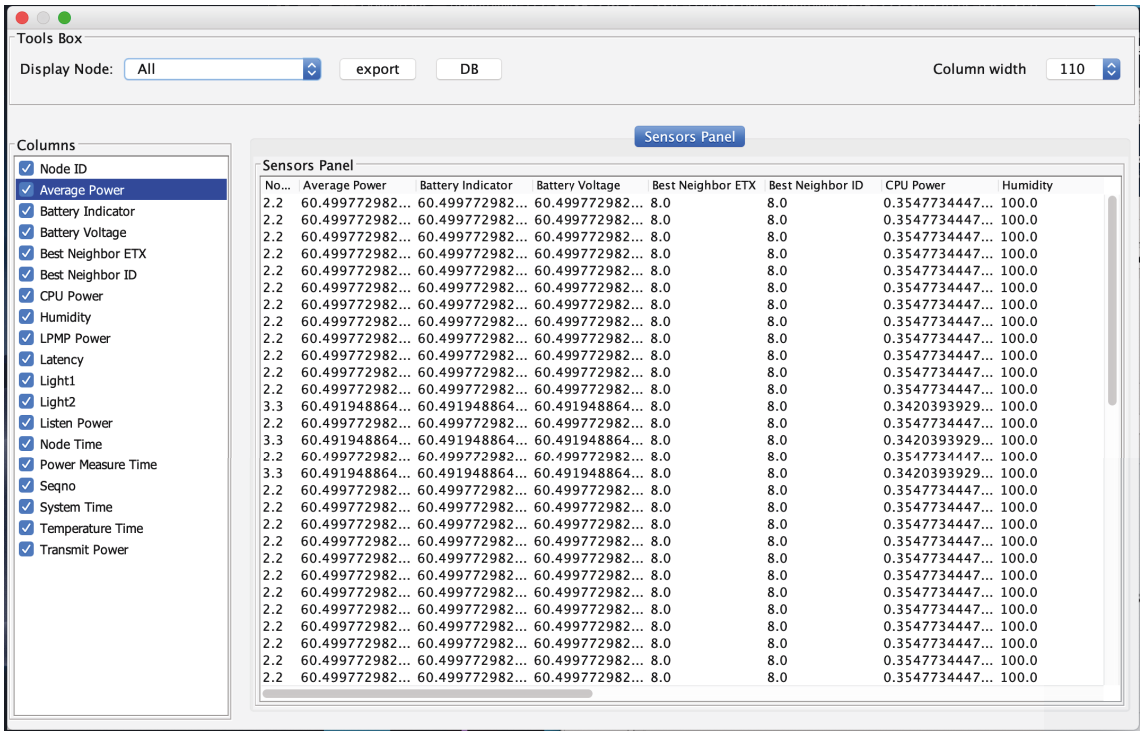


Figure 6.2 : Screenshot of the DEF tool

The DVM is the unit responsible for displaying the incoming data in a presentable format. This module allows the network administrator to monitor node performance and export the data for further processing. In order to display the sensor data, we built our custom table model by extending the Abstract Table Model class provided by Java to properly present the data in the correct form. This module was designed with user usability and convenience in mind, where the user has the ability to select the parameter of node information to display, a function not available in the original Collect View plugin. Such flexibility in-display sensor information allows the researcher to focus on specific sensor parameters. Furthermore, the module allows the user to export the data as CSV from all nodes or each node selected. Figure 6.2 shows an example of the DEF display data, with an emphasis on user convenience.

6.2.3 Database Management Module (DMM)

This module works as the database interface that handles the process of reading and writing in a database. We chose CSV and MySQL interfaces since most programming languages and tools have libraries that support both. The live data presented in the table simultaneously updates the table and the MySQL database. To achieve these functionalities, the database manager class has to be developed. This primary class function is to take user configuration and seamlessly create and update the table in the database. The Java JDBC driver [99] was utilized to achieve the connection between the DEF and the MySQL database. This allows developers to extend the capability of the collected data outside the Collect View interface and monitor the behavior of the node. One example is to utilize the MySQL [100] database to connect to a web interface that continuously monitors the nodes and their status without compromising the node power since all data is processed outside the IoT network. Moreover, since all the database processing is handled outside the IoT network's resources, this widens the spectrum of applications that can use the extracted data.

6.2.4 API Module (APIM)

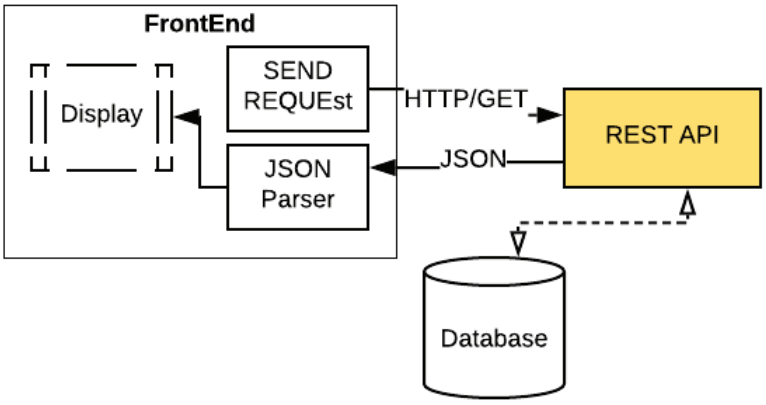


Figure 6.3 : Data exportation framework API

To simplify the process, this study proposes a simple REST API to fetch the data from the database and convert it into a data stream that can be utilized in

any third-party platform. This module is used to further enhance the developer's productivity by abstracting all the database connections and data retrieval into a simple API, ensuring good integration with third-party applications. The API comprises two main classes described below:

- **Database() Class:** The database class is used to establish a connection to the database containing all of the node's records.
- **GetData() Class:** The class is responsible for retrieving node-specific data from the database. Additionally, this class is responsible for converting the retrieved data into a JSON format to allow third-party applications to utilize the retrieved data easily.

Figure 6.4 : Example of the JSON output

```
1 [{"NodeID":"2","avgPower":"60.4598"},{"NodeID":"3","
    avgPower":"60.4319"},{"NodeID":"4","avgPower":"60.4939
    "}]
```

In the example workflow shown in Figure 6.3, a request method is used to send an HTTP request to the REST API. The API initiates a database connection to retrieve the data and send it back to the front-end. The data retrieved is then encoded in JSON format using the GetData class. Figure 6.4 shows an example of the JSON output retrieved by the DEF API. It shows the sensor's average power reading alongside the associated node ID. This JSON data structure can be utilized by any application or front-end that supports parsing JSON data supported by most programming languages.

6.2.5 DEF Evaluation

In this section, a detailed evaluation of the DEF framework is introduced. This section is divided into two parts. The first part presents the simulation design, where the design of the IoT network is discussed. The second part discusses the

evaluation of the framework by monitoring CPU and memory usage. To evaluate the DEF tool architecture, we implemented a testbed use case scenario with Cooja emulated IoT devices. The example consists of 22 nodes scattered randomly across the working area. Figure 6.5 shows the distribution of the IoT nodes. All the client nodes implement the rpl-collect.c example available in the examples folder in Contiki OS. The sink uses the UDP.sink.c. To emulate a real-world scenario where noise can be introduced in the environment, a distributor node is placed in the network area to generate signal noise in the network. Moreover, to observe how the DEF tool will perform in an attack scenario, attacker nodes are placed at the edge of the network affecting two neighbor nodes, as shown in Figure 6.5. The attacker carries a blackhole attack by advertising a better rank to the sink node. To justify the use

Table 6.1 : Workstation setup

CPU	Intel(R) i7-3740QM @ 2.70GHz
Memory	8 GB RAM
OP	Linux Contiki 4.13.0-21-generic

of the DEF tool, the evaluation scenario was executed twice with the same settings and run time except one run was without the DEF. This was done to compare CPU and memory consumption in both runs.

6.2.6 Simulation Results and Discussion

We focus on the power and memory consumption of the host computer running the DEF tool since it does not directly affect the network simulation itself. Our focus is to determine how the tool evaluates in a simulation scenario explicitly talking about two parameters: memory and CPU benchmark with and without the DEF. To evaluate each scenario's performance, we utilized the "top" command line in Linux to run every ten seconds to have a better evaluation of the current CPU and memory for each scenario.

Table 6.2 shows the final result of the evaluations. We ran the simulation six times in total — three for each scenario — with the DEF tool and without it. We observe from the table that the longer the simulation time, the less the CPU

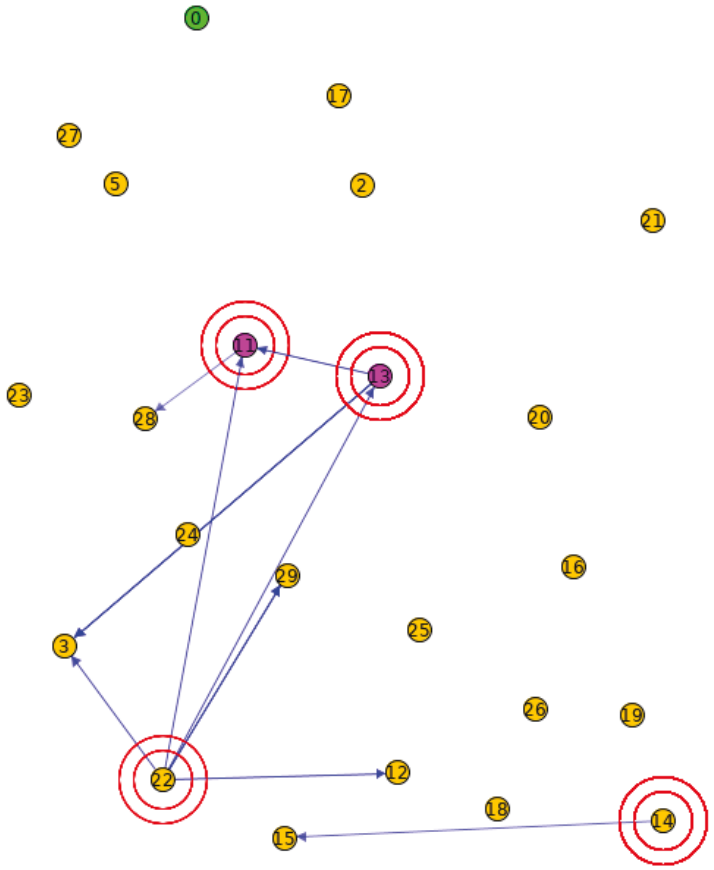


Figure 6.5 : IoT simulation network

load. This can be explained because of the caching system in the CPU. Another observation is that there is almost no change in real-time when the DEF tool is running. The result shows a minimal increase in CPU and memory usage with the DEF running in the background. This indicates that, although we are running a heavy task of data collection and representation, the DEF tool has minimal CPU and memory overhead.

The DEF tool's main contribution is to give us a clear view of the different network parameters affected by the different kinds of attacks, especially the attacks that affect the RPL networks. By using the DEF tool, we conclude that there is a set of features which need to be extracted from the RPL protocol to accurately produce a related dataset for the machine learning algorithms. In the following section, we introduce the data collection framework with all of the features explored using the

Table 6.2 : DEF Tool Performance

Scenario	AvgCPU %	AvgMemory %	SimTime(Hrs)	Realtime(S)
No DEF 1	117.56	4.4	1	49.71
No DEF 2	109.40	3.46	2	90.35
No DEF 3	103.00	6.5	3	115.44
DEF 1	118.00	5.87	1	59.71
DEF 2	110.45	4.92	2	91.15
DEF 3	107.67	6.97	3	117.22

DEF Tool.

6.3 Real-Time Data Collection Framework

Building a dataset in any context requires the careful design of the collection framework to ensure precise and unbiased results. This involves a very large effort in terms of pre-planning and how every aspect of the network should be designed, which involves a considerable amount of trial and error to ensure optimal results. In this section, we describe the process of designing the network and the different protocols involved. The network topology for this framework is shown in Figure 6.6.

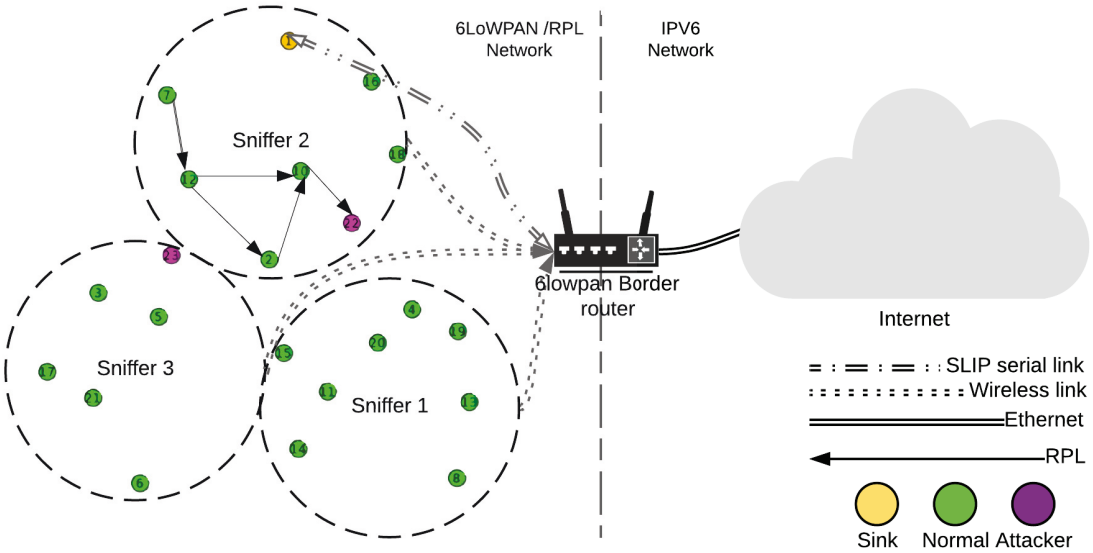


Figure 6.6 : The proposed IDS network architecture and placement

Table 6.3 : Simulation parameters

Parameter	Value
Number of nodes	29 nodes
Cluster head	1
Network area	100m x 100m
Size of packet	500
Transmission area	150m
Routing protocol	RPL
MAC protocol	CSMA
Simulation time	24 hrs

To generate the IoT-DDoS, we follow a specific procedure to ensure maximum data quality and realism. The ideal process is to have a real IoT network infrastructure in real-time operation. However, since this process is costly and can cause technical and ethical complications, we designed our network architecture to be based on a simulation/emulation design, where the IoT nodes are emulated in the Cooja environment [77].

In Cooja, the IoT nodes are emulated to have real CPU and memory power from the host workstation. Hence, when designing the framework, we considered the applicability and portability of such architecture in real-world scenarios. To emulate an IoT network's noise level, we placed two distributor nodes that emit a noisy signal at a specific interval. All of the network and simulation parameters are summarized in Table 6.3. Furthermore, the proposed system comprises four components: the capturing medium, data aggregation, queuing unit, and the feature extraction unit, as shown in Figure 6.7. In the following subsections, each part of the dataset generation process is explained.

6.3.1 Network and Attack Scenarios

The network is designed to emulate a real-world IoT network with sensors commercially available for industrial IoT. One of the popular limited resource sensors is the Zolertia Z1 mote [101], which supports the IEEE802.15.4 standard [102]. Ta-

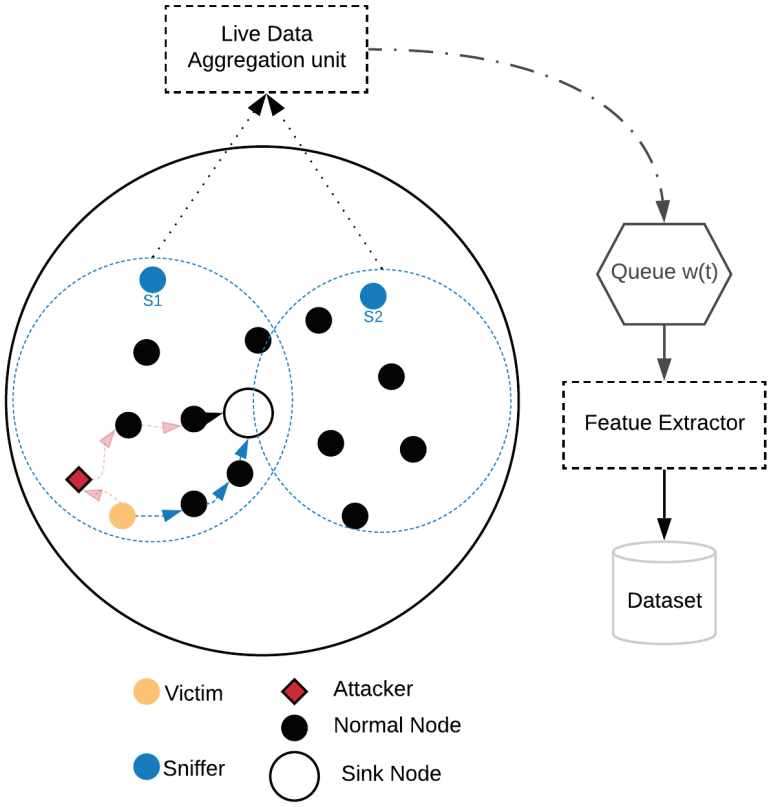


Figure 6.7 : Data collection module

ble 6.4 summarize the type of devices used and the attached sensors. We simulate distributed weather stations that collect humidity and temperature data using the SHT21/SHT25 sensors [103]

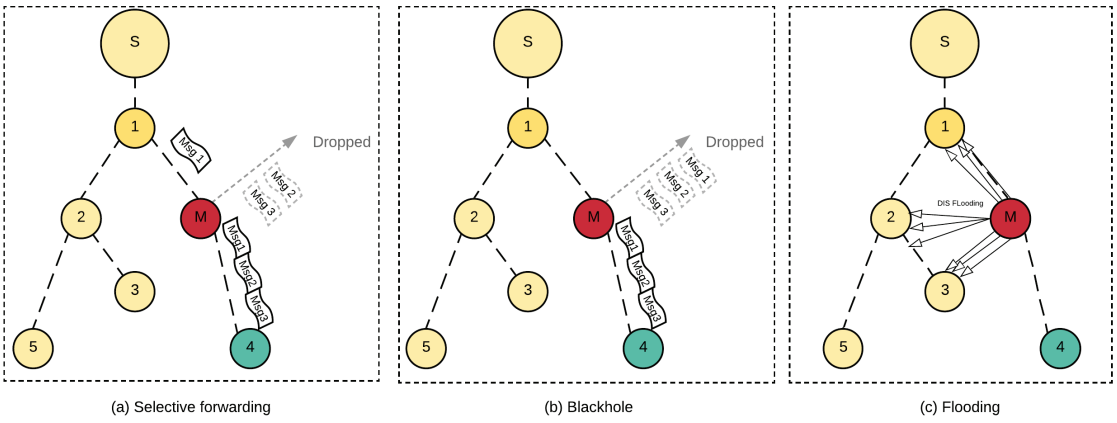


Figure 6.8 : Pictorial visualization of attacks

Table 6.4 : Sensor node parameters

Name	Value
Mote	Zolertia z1
CPU	16-bit RISC CPU @16MHz
RAM	8kb
Memory	92KB
Transceiver	CC2420
Wireless	IEEE802.15.4 / 2.4GHz
Power	3.3V/5V
Sensors	SHT21/SHT25

All of the sensors in the network use the same mote type (z1), including the sink, which means it is a homogeneous network since all of the nodes in the network use the same protocols and hardware. However, the data collection stage should not be different for a heterogeneous network that uses IEEE802.15.4 and 6LoWPAN protocols for networking [79]. All IoT nodes run on modified Contiki OS version 3.0 [104], including the sniffer nodes distributed across the network. As for routing within the 6LoWPAN architecture, we use the standard networking stack embedded within Contiki OS. However, to reproduce the three attacks, we describe the restructure the RPL implementation in section 6.3 of this chapter. The 6BR router runs natively on an Ubuntu 18.4 box that handles all of the connections coming from the sniffers and the sink node. The connection between the sink node is a serial link using the slip protocol [105]. Here we assume the sink node is a z1 node. Therefore, we use the serial link to exchange data. The connection between the Internet and the 6BR router is an Ethernet link to ensure high network throughput, in case it is needed.

To emulate a real-world IoT environment, we design four scenarios, each of which follows the same network topology. The first one is the normal behavior scenario without any attack introduced. The other three introduce some kind of attack on the network. The three variants of attacks are explained as follows:

Flooding attack: In a flooding attack, the attacker node sends an unlimited

Algorithm 3: DIS flooding attack

```
Result: Flooding attack
RplDisInterval = 0;
while i <10 do
    sendDIS();
    i++;
end
```

number of DIS messages to the victim node over a short period, causing a denial of service attack and disabling the node services. This kind of attack can also lead to battery exhaustion as it consumes all the node's resources. All of the nodes adjacent to the attacker will be affected since all these nodes connect wirelessly using the IEEE802.15.4 standard. To implement this attack in Contiki OS [104], we had to reverse engineer the networking architecture and add Algorithm 3 to the OS RPL module. The basic idea of this algorithm is to set the RPL DIS message interval to 0, and for every segment of the DIS interval, 10 DIS messages are sent continuously.

Selective forwarding attacks: This is a commonly used routing attack that tries to selectively forward only particular packets to the next node by selectively dropping specific data from the packets. Such attacks can be extremely dangerous when combined with other attacks like the sinkhole attack, leading to a Denial of Service (DoS). To implement the selective forwarding attack in Contiki OS, the networking module had to be modified using Algorithm 4, where the attacker, in its core networking module, checks if the packet is a UDP or not. If it is a UDP packet, then it is dropped, and all the other kinds of packets are sent frequently. In this type of attack, the attacker selectively drops only the UDP packets.

Blackhole attack: A blackhole attack is similar to a selective forwarding attack, but instead of forwarding specific packets, it drops all kinds of packets coming from other nodes. This attack can disturb the network topology by manipulating the ranking mechanism in the RPL protocol. To implement this attack and maximize its maliciousness, there are two steps involved. The first step is to ensure the attacker

Algorithm 4: Selective forwarding

```
Result: Drop UDP packet
packet = incoming packet;
while packets.size != 0 do
    if packet.payload == "udp" then
        drop.packet
    else
        continue;
    end
end
```

node has the best possible route to the sink node. Therefore, all of the neighbor's nodes will change their route to the sink to be through the attacker node. This is achieved by modifying the RPL ranking system in the Contiki OS source code, followed by the procedure of compiling and flashing the sensor node. The second step is similar to what has already been implemented in the previous selective forwarding attack, but instead of dropping only the UDP packets, it drops all of the packets going to the root node. This is explained in Algorithm 5.

Algorithm 5: Blackhole

```
Result: Drop all packets & and decrease rank
packets = incoming packets;
Dag Rank = root rank; while packets.size != 0 do
    if packet.dist == "root" then
        drop.packet
    else
        continue;
    end
end
```

6.3.2 Traffic Generation

Sensor traffic is generated using the simulation environment Cooja [77], where the Z1 is used with the cc2420 transceiver, as previously discussed. All the nodes run a custom humidity and temperature sensor application, which is implemented specifically for this data collection scenario. The length of the interval in which the

sensors send the data to the sink node is 30 seconds. Figure 6.10 shows a comparison of the different UDP packet flows over time between the three attacks. From the flow, we can see the blackhole attack affects far fewer UDP packets of the three attacks because the attacker drops all of the packets it receives. In the network, as can be seen from Figure 6.9, there are two types of nodes as follows:

1. **Sink node:** This is sometimes called the root node to where all the network packets are forwarded. The sink node has the highest rank across the network since it maintains the node hierarchy in the RPL network. Moreover, it is a storing type, which means all network routes are stored within this node. At the application layer level, this node works as a UDP server where it handles all of the incoming UDP packets from the client's nodes. These UDP packets contain various kinds of information, specially designed for the generation of this dataset. Furthermore, the sink node takes the responsibility of forwarding the data to the gateway node, where all of the data extraction and processing occurs. The functionality of the sink node application is developed using the core functionality available within Contiki OS.

2. **Client node:** This is a normal sensor node that senses the humidity and temperature and forwards the values to the sink node. Along with the humidity and temperature data, this node also sends other valuable information about its status, such as the RSSI level, LQI, and the ETX values encapsulated as UDP payload.

Both kinds of nodes implement the 6LoWPAN protocol on top of the IEEE802.15.4 standard for wireless communication. Although we use the IEEE802.15.4 as a communication medium for this specific solution, the proposed framework can adapt to any protocol at the physical layer level.

6.3.3 Capturing the Data

In this section, we explain the process used to capture the raw data from the IoT network. Since the IoT network usually consists of limited resource devices,

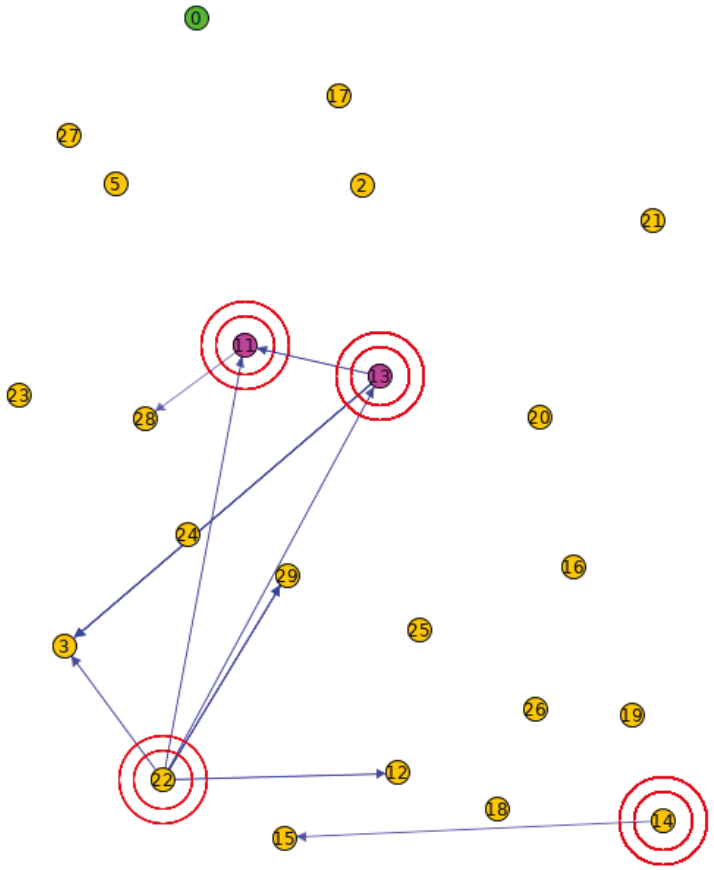


Figure 6.9 : Attack network topology

extensive data collection and monitoring is not feasible. Therefore, we propose using props placed across the network and distributed equally in the network by the network administrators. In this study, we assume the props have more power than the other nodes in the network. These props continuously monitor and collect the data and send them to the data aggregation unit. To achieve this, we use the Sensniff model [91] to sniff and monitor the network continuously. At the hardware level, the probes are equipped with the popular CC2420 [106] transceiver, which uses a large antenna to cover a wide range of the network. At the software level, Sensniff firmware is used. Sensniff is an open-source firmware used as network sniffing firmware.

On top of Sensniff, the probes are equipped with the Libpcap library [107] which is used to capture the link layer network traffic. Utilizing Libpcap, we develop a capturing algorithm defined as follows :

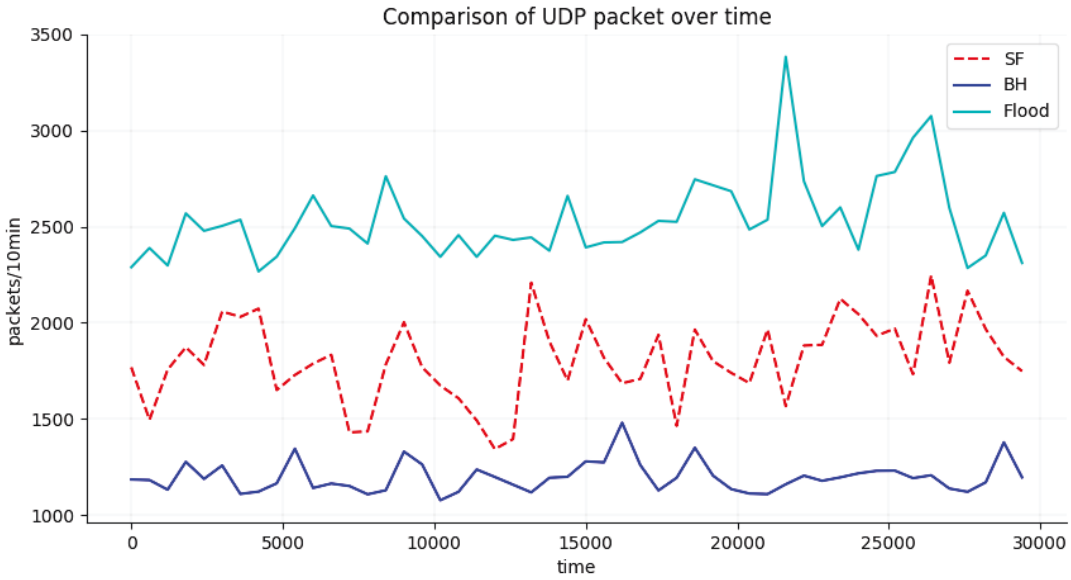


Figure 6.10 : UDP statistics

Algorithm 6: Capturing algorithm

```
Result: packet
packet = incoming packet;
while packets.size != 0 do
    tmp = packet[0].timestamp - packet.timestamp;
    if packet.size =¡ 40 then
        send packet.payload to queue;
    else
        continue;
    end
end
```

Algorithm 6 is used to define how the Libpcap library is used. Also, as can be seen, the algorithm captures the raw data at the MAC layer level to be sent later to the data aggregation unit, which is described in the next section. One critical point to mention here is that the data collected at this stage is raw IEEE802.15.4 data with all related signal information.

6.3.4 Data Aggregation

The main function of this module is to aggregate the data from multiple sources and ensure data integrity. The process is to ensure there is no data duplication

in the collected data. We create a specific algorithm embedded with the queuing algorithm as follows:

1. Check how many sniffer nodes are attached to the network and store each of these sniffers in an array. It is crucial to aggregate all of the network traffic from all the nodes across the network. Although multiple props will be needed because of the wireless signal's nature, there is a possibility of overlapping signals that can cause data redundancy and duplication. Also, more props are needed to cover the large IoT network. Therefore, the need for the data aggregation unit is crucial.

2. Iterate between each node to check if there is any node ID duplication and double-check it with the timestamp for each packet. If there is duplication as well, then run the duplication process and check by comparing all the packet signatures with the sensor node. This step is crucial since it has multiple levels of integrity check. If a packet has been duplicated, this can be identified by comparing the different attributes in each packet.

Each packet has a unique ID which is a combination of three values: the source IP, destination IP, and the timestamp. These values are used to check packet redundancy and eliminate any duplication.

6.3.5 Queue

The aggregated packet is sent to the queue system. The system takes all of the data sent from the data aggregator and checks the time window $_T$. All of the packets that are within the time window are then sent to the feature extraction unit. Alternatively, based on the number of observations, we create a variable time window, which indicates how many packets a node has sent. To shed light on this method, let us assume $_T$=10, which means for every 10 observations of data for each node extract and calculate the relevant features. Although this method might be useful, a legitimate question can be asked, this being what about the nodes that do not send anything in the network? The solution is to fill any missing data with zeros

since there is no value in this block of observations. We design a queuing method to accept either the time or number of observation windows. For this study, several observations are used as a window. Algorithm 7 shows the queuing algorithm.

Algorithm 7: Queuing algorithm

```
Sniffer: S = {S_1,...,S_n} of sizen
output: A list of queued packets
node ← list of all nodes in each S_n of size i;
while i < n do
    Q = 0 ;
    for k ← 0 to n do
        for j ← 0 to i do
            if Q > 0 then
                if node [k].pkt = node [j].pkt then node is duplicated add
                  one of them add node [k].pkt to Q ;
                else add node [k].pkt to Q;
            else Q < 0;
            featureExtraction(Q) ;
            continue
        foreach element e of the line i do FindCompress(p);
```

6.3.6 Feature Extraction Unit

The feature extraction unit is responsible for extracting the related information from the packets, as shown in Table 6.5. To ensure coverage of all three layers of the IoT communication stack, we collect the data from the three network layers since we are creating a diverse learning dataset with different features vectors. The three feature vectors are explained as follows:

1. **Feature vector 1:** This vector includes data collected at the physical layer level, such as signal strength and the transmission range.

2. **Feature vector 2:** This vector contains most of the features for this dataset where the data from the networking layer are extracted. This includes data extracted from the RPL and 6LoWPAN protocols.

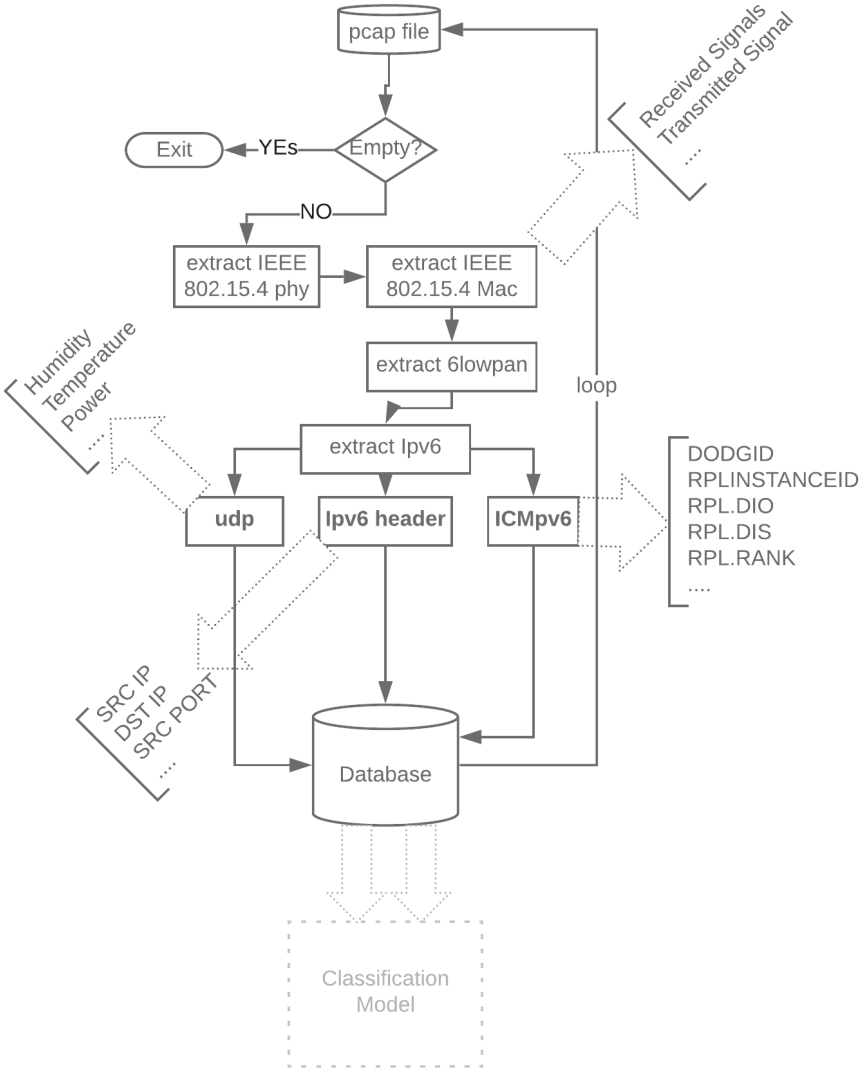


Figure 6.11 : Data collection model

3. **Feature vector 3:** This vector includes the sensor reading data extracted at the application layer, which, for our example, contains information about the humidity and temperature readings.

Moreover, most of the IoT networks run using some version of the 6LoWPAN protocol, so the related features are extracted based on the data collected, as shown in Figure 6.11. The process has several stages, as follows:

- First phase extracts the features related to the mac layer, such as the RSSI value and transmission strength. This will help the researcher to understand

the correlation of the three different attacks and the low-level layer parameters such as the transmitted and current signal strength.

- The second phase extracts the network level features such as the RPL [10] related features as described in Table 6.5. Since the blackhole attack and the selective forwarding attack occur in the network layer, these features are the most relevant to our dataset.

- The third phase is extracting the application-level features like the data payload in the UDP packet. In this layer, sensor-related information is extracted, such as battery level and humidity/ temperature information.

To extract all of this information online, we use the Scapy [108] library on Python which utilizes the powerful Linux library Libpcap for network packet capture and manipulation. The developed parser can connect to the 6BR router through a serial link port and extract all of the relevant features, as previously explained.

Our feature selection process is divided into three categories, each of which are explained as follows:

Physical Layer Features:

This is where the DCM extracts the physical layer related features such as the received and transmitted signals at the MAC layer. This information is related to physical layer jamming attacks, which interfere with the transmitted signals.

Network Layer Features:

Collecting the network layer features is crucial for our IDS since the features extracted are closely associated with many well-known attacks, such as deceased rank attacks and blackhole attacks. In this category, features such as the number of DIS and DIO messages are extracted, with several other features related to the RPL protocol. Data transmitted through any network follow specific protocols. In the IoT, there are certain protocols available at every layer, and in this study, we

Table 6.5 : Data collection features

Feature	Description
Physical Layer Features	
Received signal DBM	Mean value of the received signal at the MAC layer
Transmission signal DBM	Mean value of the transmitted signal at MAC layer
RSSI noise	Mean value of the noise recorded using RSSI
Beacon interval	Mean value of the beacon interval
Network Layer Features	
LQI	Link quality indicator
ETX	Mean value of the expected transmission count
Number of DIS messages	Number of DIS messages
Number of DIO messages	Number of RPL DIO messages
RPL rank	Number of rank changes over time.
Number of neighbors	Number of neighbours
Application Layer Features	
Temperature	Mean value of temperature
Humidity	Mean value of humidity
Power level	Mean value of energy over time
Consumed power	Mean value of consumed node power
Remaining power	Mean value of remaining node power
Node ID	Node ID

focus on the RPL protocol on top of the 6LoWPAN protocol. This can be expanded to include extra features from other protocols.

Application Layer Features:

Our module collects application-specific information at the application layer, such as the humidity, temperature, and node power level. The application related features can be extracted by programming the nodes to calculate the power consumption and other related features. We chose to collect power consumption at the application layer because we can program the node to calculate the power level and send the result within the application-related information in the UDP payload, as shown in Figure 6.12.

Including these values at the application layer allows the researcher to measure

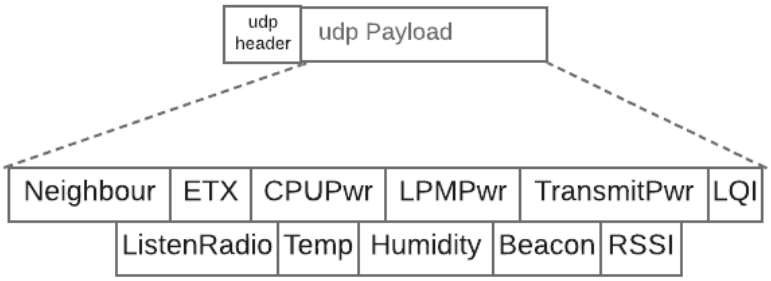


Figure 6.12 : UDP payload

how different attacks affect the sensor reading and how it can be correlated to attack detection.

6.3.7 Data Labelling

Since we are dealing with sensor networks and 6LoWPAN traffic, labeling the data based on the flow is not feasible due to the sheer amount of data transferred from each node. Furthermore, the network data in the IoT network are not transmitted as normal network flow, where TCP, for example, uses the three handshake process that can be used to extract network flow for each node. Also, since there is a connection channel between the source and destination in a traditional network flow, it can be easily extracted. Therefore, to solve these issues, we label the data based on the simulation vector and the timing. Moreover, the segment where the attack takes place will be labeled to indicate at which run the attack occurred and the type of attack. This will help the classifier to narrow down any false positive alarms. The attacker node traffic segment is also labeled to indicate that this attacker is transmitting malicious traffic. By examining the dataset, we can see that the classification method would handle blackhole attacks and flooding attacks well since there is an attack pattern, either a modified rank attack or a large number of DIS messages indicating malicious activity.

6.3.8 Quantitative Description of IoT-DDoS

The IoT-DDoS consists of 16 columns, and each column represents the nodes for which data is collected. The number of nodes captured in the network is an accurate

Table 6.6 : Protocol distribution

Protocol	Packets	Size(Bytes)	(%)
IEEE 802.15.4	1391817	289.5471874	33.17%
6LoWPAN	1362111	299.0477588	32.47%
UDP	553529	15.78560291	13.19%
ICMPv6	888080	93.6091151	21.17%

representation of our proposed capture method. Figure 6.14 shows how frequently the packet was sent across the network in a normal scenario. Table 6.6 shows the contribution of each protocol in the dataset. This is the overall percentage for all of the four scenarios executed. However, this percentage changes when compared individually for each attack. For example, the number of UDP packets for the blackhole and flooding attack drops significantly compared to the ground truth normal behavior dataset.

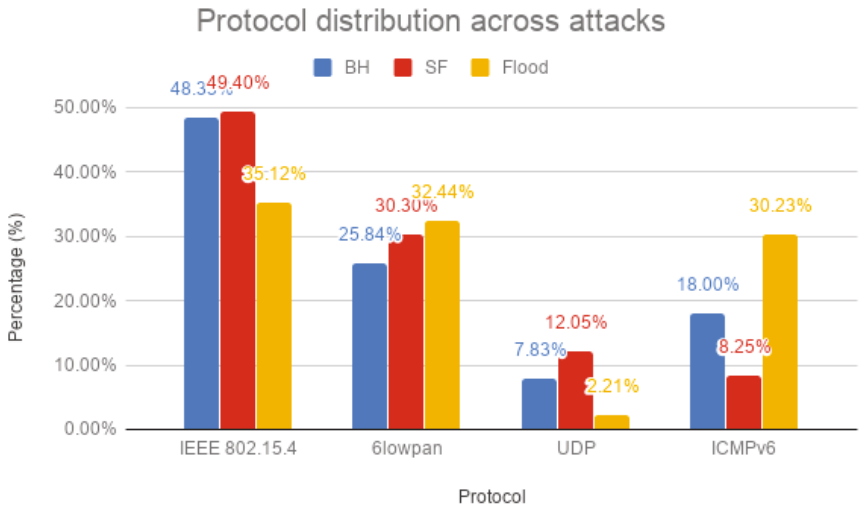


Figure 6.13 : Protocol distribution

On the other hand, the flooding attack generates the highest number of DIS messages compared to all the other scenarios. Figure 6.13 shows the protocol distribution from the attack perspective. Table 6.6 shows the percentage and size of each protocol in the whole dataset generated. One thing to note is that the results

from all of these files are four network PCAP traces that are used to extract these results and generate the dataset.

6.4 Analysis of the Dataset

The IoT-DDoS dataset contains various attack structures as previously discussed. To provide insight into the complexity of this dataset, we analyze the 96 simulated hours of traffic using all of the acquired features in the different network layers.

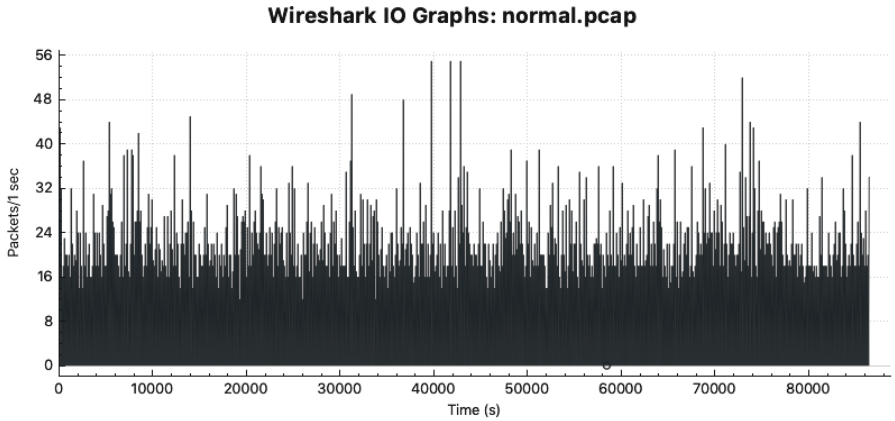


Figure 6.14 : Network flow

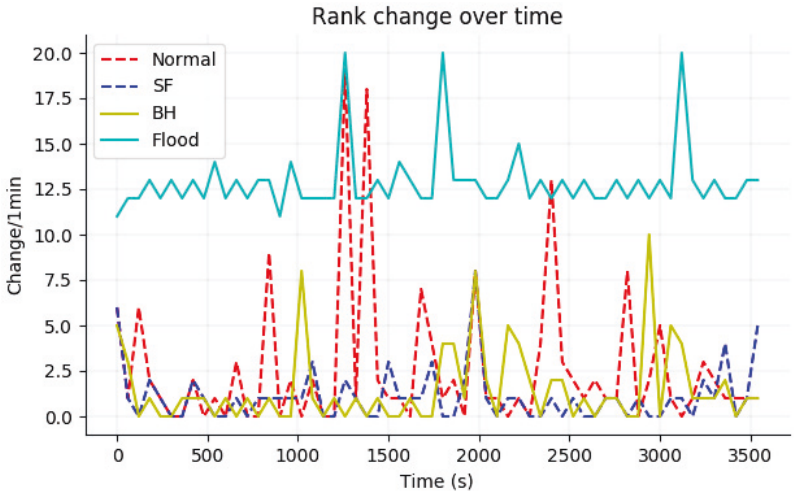


Figure 6.15 : Rank change over time

Figure 6.14 shows the network flow of the ground truth normal behavior scenario.

One critical aspect of the dataset we want to analyze is how the RPL protocol performs under such long intensive attacks. Figure 6.15 shows the rank change over time of one of the parent nodes that sits at the end of the DoDAG tree. It can be seen that the average of the rank changes over time is extremely high in the case of flooding attacks in comparison to the ground truth normal behavior. This can be explained because of the type of flooding we implemented, that is, an RPL DIS flooding attack where the attacker node sends unlimited RPL DIS messages in a short interval. Interestingly, this attack affects the ranking behavior of the nodes across the network. Another observation is how the blackhole rank changes for this specific node. As can be seen, it is the most stable among all of the scenarios. The reason is that the attacker was set to have the best rank in the network during the simulation period. On the other hand, the selective forwarding attack shows a similar network pattern to the normal behavior scenario in terms of DIS and DIO messages. However, there is a small abnormal observation when we look at the rank changes over time.

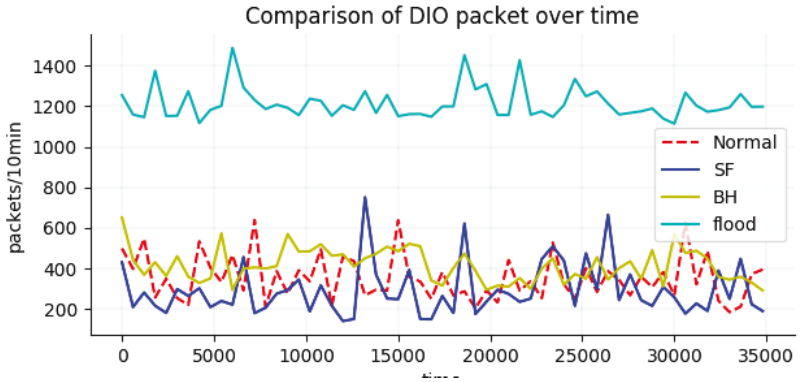


Figure 6.16 : DIO comparison

There is a frequent fluctuation in the node rank, and we think this is due to the stealthiness of the selective forwarding attack and the way it can be misdiagnosed as normal behavior. One important aspect which must be considered when analyzing the RPL protocol is the number of DIO messages sent by the node over time. Figure 6.16 shows the DIO messages from each scenario, and as can be seen, the flooding attack has the highest number of sent DIO messages, which we explained earlier as

Table 6.7 : Sample of the dataset

Node ip	Window	ReceivedDBM	TransmitDBM	RSSI	BeaconInterval	LQI	NumOfDIS	NumOfDIO	Temperature	Humidity	PowerLevel	ConsumedPower	RemainingPower
ffff:0000:0001	1	5.68826E+12	0.014880723	16	0	65449	0	4	615	2650.1675	37	0.039230499	0
ffff:0000:0002	1	1.2712E+13	0.060292862	20	0	32717.5	0	2	615	2650.1675	37	0.101110124	0
ffff:0000:0003	1	1.2712E+13	0.060292862	20	0	32717.5	0	2	615	2650.1675	37	0.101110124	0
ffff:0000:0004	1	1.2712E+13	0.060292862	20	0	32717.5	0	2	615	2650.1675	37	0.101110124	0
ffff:0000:0005	1	1.2712E+13	0.060292862	20	0	32717.5	0	2	615	2650.1675	37	0.101110124	0
ffff:0000:0006	1	2.17455E+13	0.050235145	16	0	32716	0	2	615	2650.1675	18.5	0.059058167	0
ffff:0000:0007	1	1.12759E+13	0.035503919	16	0	65441	0	2	615	2650.1675	37	0.070549923	0
ffff:0000:0008	1	1.12759E+13	0.035503919	16	0	65441	0	2	615	2650.1675	37	0.070549923	0
ffff:0000:0009	1	3.11313E+13	0.031342354	16	0	65414	0	4	615	2650.1675	37	0.061673546	0
ffff:0000:0010	1	3.11313E+13	0.031342354	16	0	65414	0	4	615	2650.1675	37	0.061673546	0
ffff:0000:0011	1	1.54464E+13	0.039935698	32	0	65426	0	5	615	2650.1675	37	0.039900808	0
ffff:0000:0012	1	3.41581E+13	0.044062105	16	0	65436	0	1	615	2650.1675	37	0.064181777	0
ffff:0000:0013	1	2.41675E+13	0.038715397	32	0	21809.33333	0	8	615	2650.1675	37	0.058440722	0
ffff:0000:0014	1	2.41675E+13	0.038715397	32	0	13085.6	0	9	615	2650.1675	37	0.058440722	0
ffff:0000:0001	2	3.75446E+13	0.023274421	16	0	65420	0	1	615	2650.1675	0	0.03352426	0
ffff:0000:0002	2	3.75446E+13	0.023274421	16	0	65420	0	1	615	2650.1675	0	0.03352426	0
ffff:0000:0003	2	3.52057E+13	0.227608124	32	0	119	0	6	615	2650.1675	37	0.233043085	0
ffff:0000:0004	2	3.52057E+13	0.227608124	32	0	119	0	6	615	2650.1675	37	0.233043085	0
ffff:0000:0005	2	3.52057E+13	0.227608124	32	0	119	0	6	615	2650.1675	37	0.233043085	0
ffff:0000:0006	2	3.12032E+13	0.109457518	32	0	65410	0	5	615	2650.1675	37	0.124687374	0
ffff:0000:0007	2	4.30547E+13	0.029680223	32	0	65420	0	5	615	2650.1675	0	0.048699772	0
ffff:0000:0008	2	5.85994E+13	0.116518456	32	0	65415	0	4	615	2650.1675	37	0.135271402	0
ffff:0000:0009	2	5.85994E+13	0.116518456	32	0	65415	0	4	615	2650.1675	37	0.135271402	0
ffff:0000:0010	2	6.01304E+13	0.033911444	32	0	65421	0	3	615	2650.1675	37	0.108536075	0
ffff:0000:0011	2	6.16961E+13	0.044299193	32	0	122	0	4	615	2650.1675	0	0.055924893	0
ffff:0000:0012	2	6.53184E+13	0.095108854	32	0	32704.5	0	4	615	2650.1675	37	0.120918141	0
ffff:0000:0013	2	4.33398E+13	0.069514919	32	0	65413	0	4	615	2650.1675	37	0.099507365	0
ffff:0000:0014	2	4.33398E+13	0.069514919	32	0	65413	0	4	615	2650.1675	37	0.099507365	0

being due to the frequency of the attack. However, there is a considerable difference between the blackhole attack and the normal behavior scenario. This is because the rank value is encapsulated in the DIO message; therefore, a higher number of packets is sent. Table 6.7 show a sample of the generated dataset.

6.5 Comparison With Other Datasets

In this section, we compare some of the characteristics of the IoT-DDoS to other available datasets. Although some of these datasets were considered the gold standard for the research community in the area of IDS evaluation, to date, no dataset has been designed to adapt to the emerging IoT for security evaluation. This research can be differentiated from the prior work by its capability to address the limitations associated with the previous datasets.

Table 6.8 : Dataset comparison

Dataset	Realistic network	Labeled	Protocols				Attributes	IoT attacks	Type of flow	IoT environment
			TCP/IP/UDP	6lowpan	RPL	IEEE802.14.5				
KDDcup99	Yes	Yes	Yes	No	No	No	41	No	Dataflow	No
Darpa	Yes	Yes	Yes	No	No	No	41	No	Dataflow	No
NSL-KDD	Yes	Yes	Yes	No	No	No	-	No	Dataflow	No
Koyoto	Yes	Yes	Yes	No	No	No	-	No	Dataflow	No
CAIDA	Yes	Yes	Yes	No	No	No	-	No	Dataflow	No
UNBIS	Yes	Yes	Yes	No	No	No	-	No	Dataflow	No
TUIDS	Yes	Yes	Yes	No	No	No	-	No	Dataflow	No
Sperotto	Yes	Yes	Yes	No	No	No	-	No	Dataflow	No
MAWILab	Yes	Yes	Yes	No	No	No	-	No	Dataflow	No
IoT-DDoS	No	Yes	Yes	Yes	Yes	Yes	12	Yes	Time , Packet based	Yes

Table 6.8 summarizes all the explored datasets and compares them with the

IoT-DDoS datasets, which is the result of this chapter. As can be seen from the table, most of the datasets were designed for different sets of protocols that are not currently available in today's IoT networks. No datasets have been collected for network-level IoT data, (e.g., 6LoWPAN and RPL traffic) which works as the base of many IoT communication technologies in the market today. Furthermore, none of the explored datasets was designed with IoT malicious activity in mind; all of the datasets evaluated the risk activity of a TCP/IP network. In the IoT-DDoS, we were able to implement three kinds of IoT-related attacks, as explained in this chapter. This can be used as the basis to implement a new solution to counter these attacks using artificial intelligence. Table 6.8 details the different parameters involved in creating each dataset, showing that no dataset was explicitly designed for IoT network protocols. Instead, the focus was on a TCP/IP based traditional network. In the IoT-DDoS dataset, 16 features were extracted from the three-level layers.

6.6 Conclusion

A concerns of researchers in the computer science field is the availability of datasets in their specific area. However, despite the research community's effort in various areas of computer science, there are still limitations when it comes to datasets designed for IDS evaluation and testing. Hence, the need for dynamically updated datasets is crucial to ensure the maximum interoperability across different emerging technologies.

We also shed light on the issues associated with previous datasets when it comes to IoT security applications. Furthermore, we addressed some of these issues by designing a systematic approach for dataset creation in IoT ecosystems. The result was the generation of the IoT-DDoS, which includes implementing three different attacks related to IoT security. The applicability of this dataset can be extended to include more attacks and security issues. However, at this stage, this dataset addresses the need for a comprehensive dataset for IoT security research with three popular attack scenarios. The significance of this dataset to the existing literature is

two-fold as follows: (a) this dataset provides IoT researchers with a relevant DDoS dataset that they can use to validate the models developed to counter DDoS attacks; (b) this dataset can be regenerated by the users since the developed framework can be used to collect data at any point in time. Hence, this dataset is self-sustaining in this regard. We explored all of the available datasets and compared them to our dataset to ensure the feasibility of the new dataset generated. In the next section of this thesis, we utilize the dataset generated in this chapter to build a machine learning model for IoT attack detection.

Chapter 7

A Machine Learning Platform for Detecting DDoS Attacks in IoT Based Networks

7.1 Introduction

The Internet of Things (IoT) is already a part of the infrastructure of many sectors that are involved in our daily life. However, security is playing an important role when it comes to the speed of IoT deployment. One way to protect IoT devices and networks is by building a machine learning infrastructure responsible for detecting and preventing networks and devices from malicious attacks. In Chapter 2 of this thesis, we explored the literature and found there is no machine learning infrastructure to detect DDoS attacks in IoT networks. In the previous chapters, we presented a framework for building a machine learning IDs for IoT networks. This includes a data collection framework to acquire the dataset to train the machine learning model. Building on the previous chapters of this thesis, in this chapter, we describe a machine learning model using the dataset gathered in Chapter 6 Furthermore, we develop three algorithms, specifically tailored for IoT networks, to detect three kinds of attacks. The main contribution of this work can be summarized as follows:

1. First, we summarize the machine learning mathematics in relation to our framework.

2. Second, we thoroughly analyze the three machine learning methods and investigate the best parameters for different scenarios under three kinds of attacks.

3. Third, there are multiple issues with detecting novel DDoS attacks due to their unpredicted behaviors with different attack patterns, making the detection of such attacks a challenging task. Therefore, in this chapter, we propose an

SVM machine learning method tuned to detect IoT DDoS attacks, mainly blackhole, selective forwarding, and flooding attacks.

This chapter is divided into sections described as follows: Section 7.2 explores the three different machine learning algorithms theoretical details of the in relation to our system. In section 7.3, the implementation and testing of the machine learning algorithms are presented. In Section 7.4 the implemented machine learning methods are evaluated. In section 7.5 a comprehensive comparison and discussion of the experiment results and the strengths of each one are investigated. Finally, section 7.6 concludes this chapter.

7.2 Machine Learning Algorithms for Attack Detection

In this section of this chapter, we explore three machine learning algorithms and explain all the related features and parameters, which will help us to understand how each machine learning model behaves against our datasets. This thesis explores three popular machine learning algorithms that have been widely used for attack and anomaly detection, support vector machines, decision trees and neural networks, which are explored in the following sub-sections:

7.2.1 Support Vector Machine Testing

A support vector machine (SVM) is a machine learning algorithm that has been widely used for IDS; many applications in the literature have applied SVM for different security-related applications [109] [14] [110] [111]

Cortes and Vapnik developed the current SVM algorithm [112] in 1995. It is a machine learning algorithm that is used both for regression and classification by taking the training vector as an input and trying to separate the data into classes based on the n-dimension features. The boundary that separates different classes is called the hyperplane, which separates the data points into different classes based on their features. However, in a single or multidimensional dataset, multiple hyperplanes exist [109]. Therefore the job of SVM is to identify the best and most optimal hyperplane margin for classification. Figure 7.1 shows the hyperplane. To

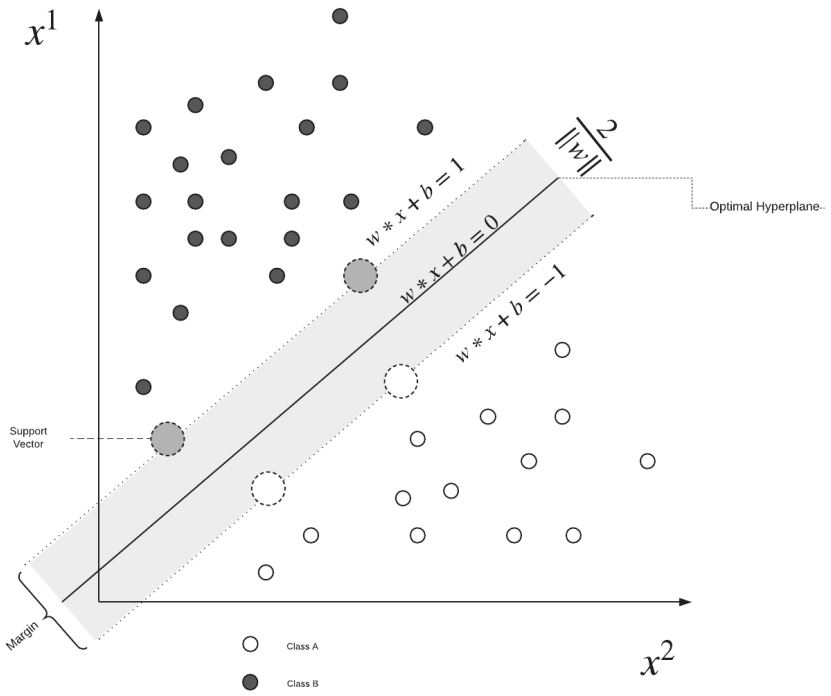


Figure 7.1 : SVM diagram

successfully classify the data points, the algorithm has to maximize the distance between the hyperplane and the data point using the hinge loss function defined as follows:

The following mathematical equation represents the hyperplane:

$$w.x + b = 0 \tag{7.1}$$

where b is the bias term, x is the input data point vector and w is the variable weight. For each data point, there are two possibilities: 1 which means that the value is part of the (+) class, alternatively, -1, which means the data point is part of the (-) class.

$$\begin{cases} w.x + b = 1 \\ w.x + b = -1 \end{cases} \tag{7.2}$$

To elaborate this further, let us assume we have a set of data $x_{i \in R^{d},\ i = 1,,,,t}$

where x represents the data point and superscripts with (i) to indicate the appearance of that instance, assuming that x_i is part of either of two classes $y_i \in \{1, -1\}$. For these two classes, in our case where we try to detect anomalies in IoT network traffic, we can say that the 1 class represents the normal traffic and the -1 represents the traffic anomalies. Therefore in relation to what is discussed in equation 7.1 and 7.2 this can be rewritten as follows:

$$\begin{cases} w^T \cdot \ x_i + b \ \geq 1 \ for \ all x_i \in Normal \\ w^T \cdot \ x_i + b \ \leq -1 \ for \ all x_i \in Anomaly \end{cases} \tag{7.3}$$

If the set can be separated linearly, this is expressed as follows:

$$y_i(w^T \cdot x_i + b) \ \geq \ 1 \ for \ all \ i = 1, ...L \tag{7.4}$$

If we refer to 7.1, the dotted line that separates the hyperplane is called the margin, which is the distance between the support vector from both classes, and it can be calculated by obtaining the norm of w as follows $\frac{2}{\|w\|}$. This can be written as minimizing the $\|w\|^2$, similarly as maximizing the distance of the hyperplane between the classes. Therefore, the main problem that SVM is trying to solve is to find the optimal hyperplane. This can be achieved by maximizing the margin between the hyperplane and the two classes, as previously stated, which can be represented by the following equation:

$$\mathbf{min}(w) = \frac{1}{2} \ \|w\|^2 \tag{7.5}$$

$$\mathbf{subject\ to} \ \ y_i(w^T \cdot \ x_i \ + b) \geq 1 \ , \ i = 1, ..., l. \tag{7.6}$$

The above equation is a quadratic optimization problem, which is the main mathematical problem the SVM algorithm tries to solve. This optimization problem is best used with sets that have a clear separation between classes, which can be

separated linearly. However, for complex sets that cannot be separated linearly (which is the case for the data collected from IoT networks), it will be difficult for such a problem to classify them. Therefore, to solve this issue, the Lagrange [113] problem and Wolfe dual [114] problem are used. The Lagrange representation for the SVM is as follows :

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w \cdot w - \sum_{i=1}^{m} \alpha_i [y_i (w \cdot x + b) - 1] \tag{7.7}$$

where α is the Lagrange multiplier, and the aim is to maximize it for each data-point instance x_i, there is also w, b for each data point, and we need to minimize the $\frac{1}{2} \|w\|^2$

This is sometimes called the hard margin SVM [115] which works well with linearly separated data points. In a real-world scenario where network traffic exists, there is a considerable amount of noise in the dataset. Therefore it is hard to separate them using the hard margin SVM. This issue can be addressed using the slack variable, which is more flexible when the objective functions want to meet the constraints. The new constraint with the slack variable is as follows:

$$y_i (w \cdot x_i + b) \geq 1 - \zeta_i, i = 1...m \tag{7.8}$$

Therefore the new equation after the slack is introduced is :

$$\textbf{min } w, b, \zeta \frac{1}{2} ||w||^2 + \sum_{i=1}^{m} \zeta_i \tag{7.9}$$

$$\textbf{subject to } y_i (w \cdot x_i + b) \geq 1 - \zeta_i, i = 1...m \tag{7.10}$$

However, a problem occurs when choosing a large value for the slack variable, which is similar to the hard margin SVM, causing the SVM to be unforgiving and failing to satisfy all the constraints. To solve this issue, the C parameter is used, which is also called regularization. Through equalization using the (C) parameter, the ζ is more manageable, since specifying a small (C) value will emphasize the

importance of ζ and a larger (C) value indicates no importance to the ζ variable. The new equation after adding the C parameter is :

$$\textbf{Min} w, b, \zeta \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\zeta_i \tag{7.11}$$

$$\textbf{subject to } y_i(w \cdot x_i + b) \geq 1 - \zeta_i, \zeta_i \geq 0, i = 1...m \tag{7.12}$$

If we substitute Equation (7.11) using the dual problem in the Lagrangian function the result will be as follows :

$$\textbf{Max } \alpha \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i \cdot x_j \tag{7.13}$$

$$\textbf{Subject To } \leq \alpha_i \leq C, i = 1...m, \sum_{i=1}^{m} a_i y_i = 0 \tag{7.14}$$

This is the equation that SVM is trying to solve to draw the hyperplane and separate the data points. Next, we discuss the type of kernels used by SVM.

Kernel

In the previous section, we discussed the use of a slack variable and the C parameters to separate data when there is noise which prevents it from being linearly separable. However, what if the data without noise is not linearly separable?. This means the data by design cannot be separated linearly. This can be solved using what is called the kernel trick. The kernel trick in its basic forms solves the dot product of $x_i \cdot x_i$ in the dual optimisation problem which can be defined as $K(x_i, x_j) = x_i \cdot x_j$ and substituted in the dual optimization problem as follows :

$$\textbf{Max } \alpha \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i \cdot x_j) \tag{7.15}$$

$$\textbf{Subject To } \alpha_i \geq 0, i = 1...m, \sum_{i=1}^{m} a_i y_i = 0 \tag{7.16}$$

There are different kinds of kernels that can be used in SVM algorithms which are explained as follows:

- The linear kernel, which is defined as $K(x_i, x_j) = x_i \cdot x_j$, is usually used for text classifications since it produces good results.
- The polynomial kernel which is defined as $K(x_i, x_j) = (x_i \cdot x_j + c)^d$, with the C parameters and d for more flexibility.
- The radial basis function (RBF) kernel which is defined as $K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2)$, is one of the popular kernels used with SVM which uses γ to define how the kernel should fit the data. A small value of γ will cause the model to under-fit and behave like a linear model. A large value of γ will make the model overfit with good accuracy, but it does not generalize well.

This is one of the algorithms that will be used in this thesis with the RBF kernel, as discussed in section 7.3 of this chapter. In the next subsection, the neural network and random forest method will be explored and discussed in detail.

7.2.2 Artificial Neural Networks

Neural networks have been used for security applications [116][117] for a long time and can be applied in different sectors of security in IoT. In this section, we explore one kind of ANN called the multi-layer perception [118] to detect different kinds of attacks based on anomaly detection.

The neural networks algorithm is considered to be one of the best algorithms for prediction and classification [119]. The reason for this is apparent as it reflects the

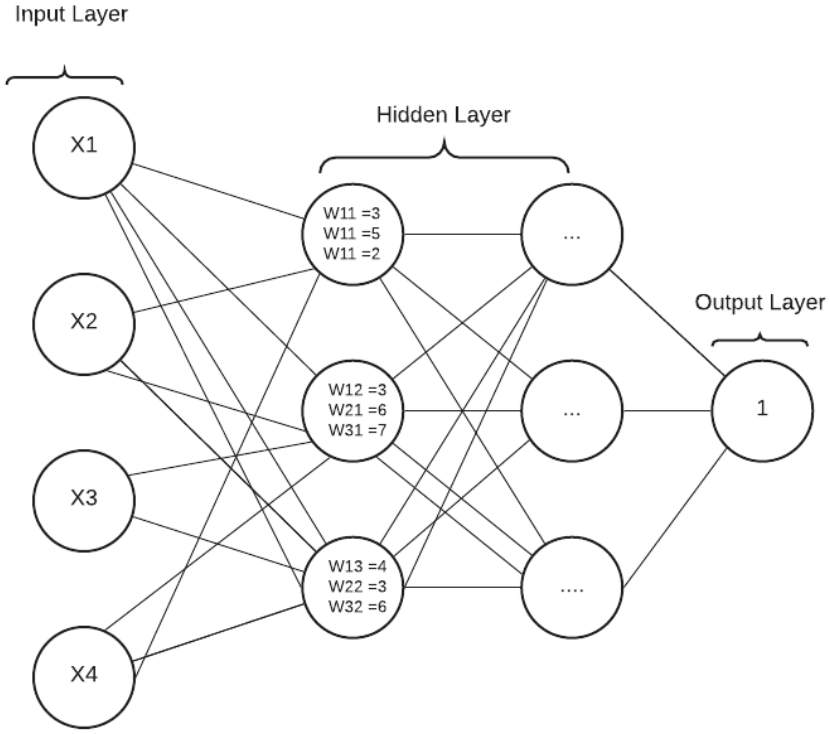


Figure 7.2 : Multi-layer perceptron

human neuron system that helps humans think and make daily life decisions. The neural network structure is a layered structure with the input and hidden layers in multi-layered perceptron (MLP). A simple MLP is called a single-layer perceptron neural network. It is a neural network with one input layer and one output layer, which can be used for simple classification. However, it does not support in-depth classification where multiple hidden layers are introduced for complex processing.

The number of attributes defines the input layer, which represents the dataset number of features or classes. The output layer has the input/2 output, each neuron has some numeric value assigned called weights, and the input of a data tuple is called the input values (features). To proceed, we must calculate the next neuron's output value, which can be calculated by multiplying the weight and the input value, which results in the following equation 7.17

$$output = input * weight + bias \tag{7.17}$$

which is considered as the value of the next neuron's output or the output layer's value. However, depending on the input neurons, we calculate all the input values, and the largest value is considered as the output of that neural network's output. The general formula for a single perception is:

$$output = b \cdot x_i \cdot w_i \tag{7.18}$$

There is another type of neural network that gives a more precise output result known as the multi-layer perceptron neural network (MPL), as shown in Figure 7.2. It represents the structure of the human neural network more accurately. It introduces the hidden layer concept that further provides a classification of attributes. The number of hidden layers depends entirely on the flexibility of data or the volume of data. The greater the number of hidden layers, the more precise the results will be.

Weight

As can be seen from Figure 7.2, we have three input neurons fed into a hidden layer with four neurons. Each hidden layer is represented by W_n where n is the number of hidden layers. We can also see that each connection between neurons has the weight ex($w31 = 7$). If we look at the input $x2$ which connects to the neurons $w31$ which have the highest value of 7, the neurons in this case think that $X2$ is the most important feature from the other features since it has the highest weight.

Bias

The other value in the neural network is bias. Bias, in simple terms, is another kind of weight assigned to the hidden layer neurons. This is used with the weight to modify the output and both of them help the neural network model to precisely fit the data to obtain the best result possible.

Activation

Each neuron contributes to the neural network by providing a small decision that affects the final output value. This process is called the activation function, and the process of aggregating all of those small contributions is represented by z. Therefore, the function looks like this $F(z)$; in this case, it is a linear function. Another activation function exist, but for this thesis we use the Sigmoid activation function which is represented by the following equation:

$$f(x) = 1/(1 + e^{(} - 1 * z)) \tag{7.19}$$

In the next section, we discuss the mathematics behind the random forest algorithm and how it works.

7.2.3 Random Forest

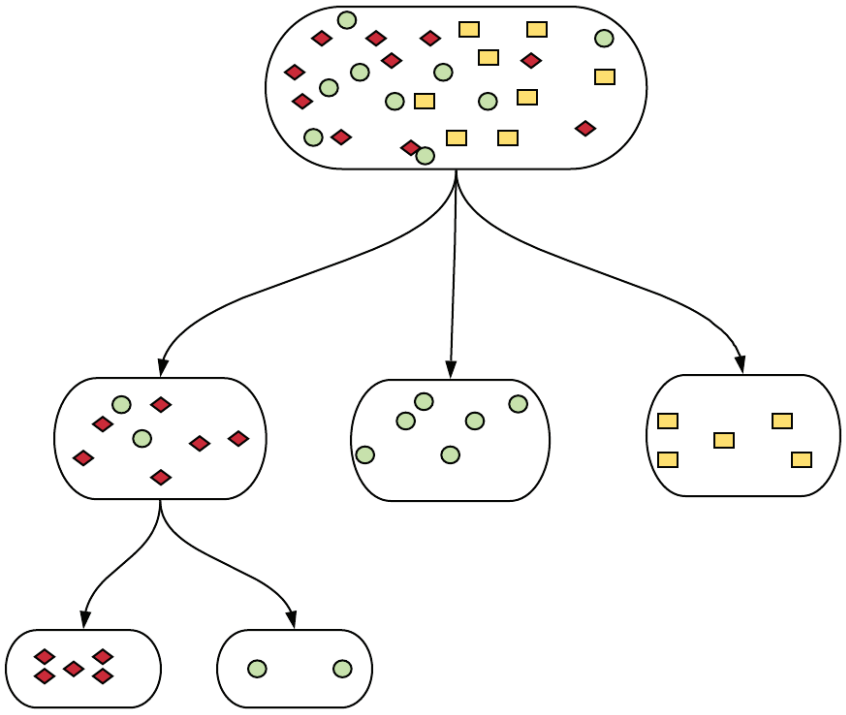


Figure 7.3 : Random forest example

A decision tree is an old method used for both classification and prediction in

machine learning. The concept is based on tree topology. In a decision tree, each point is called a node. The first node or the starter of the tree is called the root node, and the last are called leaves. The tree progresses with a binary decision making sequence. Each node is connected to two new nodes that represent the state of true and false, respectively. This is achieved by splitting the data into different classes based on their features. This is done by choosing the splitting method. This thesis uses a random forest [120] with the Gini impurity [121] splitting method. Furthermore, for testing and comparison, the entropy splitting method is also used.

The random forest is one kind of decision tree that randomizes the process of building a tree in the forest. It achieves this by randomizing the training samples and also by randomly selecting different subsets of features when the new tree is generated.

The Gini-impurity method used for splitting can expressed mathematically as follows:

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2 \tag{7.20}$$

This process is recursively repeated until it reaches the maximum depth of the tree, which is defined in Python as $Maxdepth$. Otherwise, it will stop when there is only one sample for that class. Different parameters of the random forest model are modified in the implementation to achieve the best result possible. This model is discussed in detail, including the different parameters in the next section.

7.3 Machine Learning Pre-Processing

This section outlines the process followed to train, test, and validate the machine learning algorithms The pre-processing phase focuses mainly on the preparation of the three machine learning methods using the IoT-DDoS dataset. The three methods (SVM, random forest, neural network)are implemented and analyzed using the Sklearn [122] framework. The RBF kernel is used for the SVM method since it produces the most relevant result when applied to our dataset. Before diving deeper

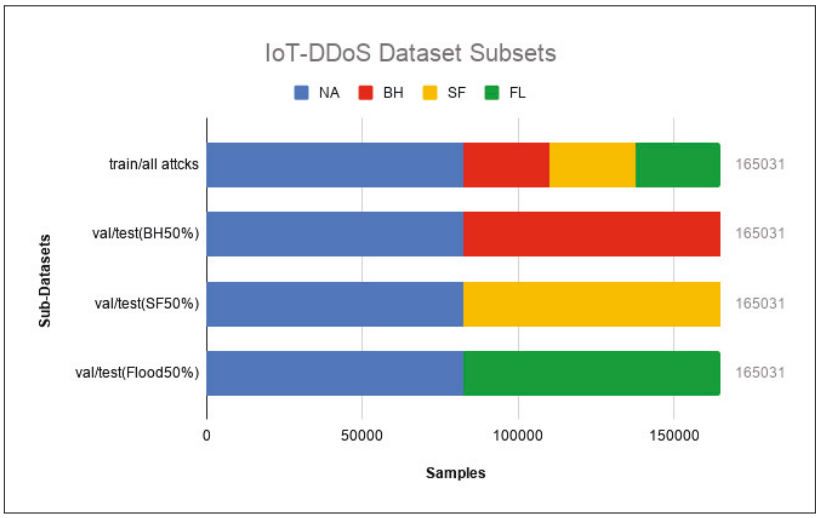


Figure 7.4 : IoT-DDoS subsets

into the analysis of the machine learning result, we must understand that three main steps are commonly involved in machine learning applications: the feature selection, training phase, and validation/testing phase. The validation phase is crucial since there is a golden rule when applying any machine learning method which is to refrain from using the training data for testing to avoid biased results. Therefore, for our verification and testing, we divide the dataset into four sub-datasets, as shown in Figure 7.4, which are described as follows:

1. Training dataset: consisting of the dataset of 50% of the no-attack dataset and 50% of the all-of-the-attacks divided equally as 16.6 % for each attack

2. Validation/testing (selective attack): consisting of 50% of the training dataset and 50% of selective forwarding attack.

3. Validation/testing (blackhole attack): consisting of 50% of the training dataset and 50% of blackhole attack.

4. Validation/testing (flooding attack): consisting of 50% of the training dataset and 50% of a flooding attack.

These datasets are further divided and shuffled using the k-fold validation method to ensure that none of the training data is leaked into the testing phase. Figure 7.4

shows the number of k-folds and the dataset division process.

Once all of the sub-datasets are prepared, the primary training dataset and the validation sub-datasets are used to evaluate and tune the parameters of the machine learning methods. After the best results are acquired in the validation/training phase, the final results are displayed in this thesis and discussed extensively in section 7.5 of this chapter.

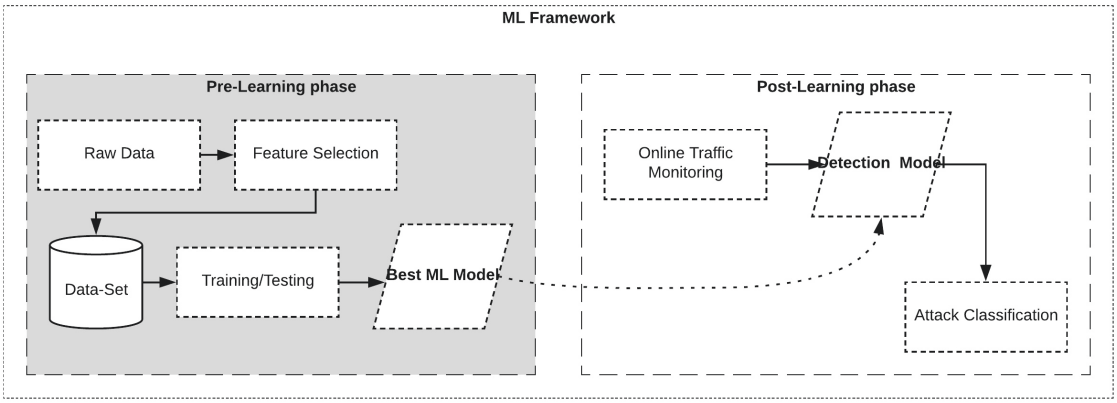


Figure 7.5 : The ML framework used in this thesis

We use the following framework to evaluate the three machine learning algorithms. As can be seen in Figure 7.5, we are in the pre-learning phase where the process is divided into three sub-steps which are discussed in detail in the following sections:

Feature Selection

As previously discussed in the introduction of this chapter, this chapter evaluates the performance of different supervised machine learning methods under different attack circumstances using the IoT-DDoS dataset generated in Chapter 6. The IoT-DDoS was explained extensively in chapter six, including all of the feature selection processes. The dataset contains three different features of vectors extracted from the three IoT network stacks. In this thesis, this process is carried out by analyzing the behavior of different network parameters under different attack scenarios. This

Table 7.1 : Dataset samples

Attacks	Samples
Selective Forwarding	193890
Blackhole	211366
Flooding	229898
No Attack	165031
Total	800185

process was explained in section 4 of chapter 6, specifically in the dataset pre-collection phase. Figure 7.6 shows the correlation of each feature in the dataset and Table 7.1 shows the number of samples per attack on the IoT-DDoS dataset.

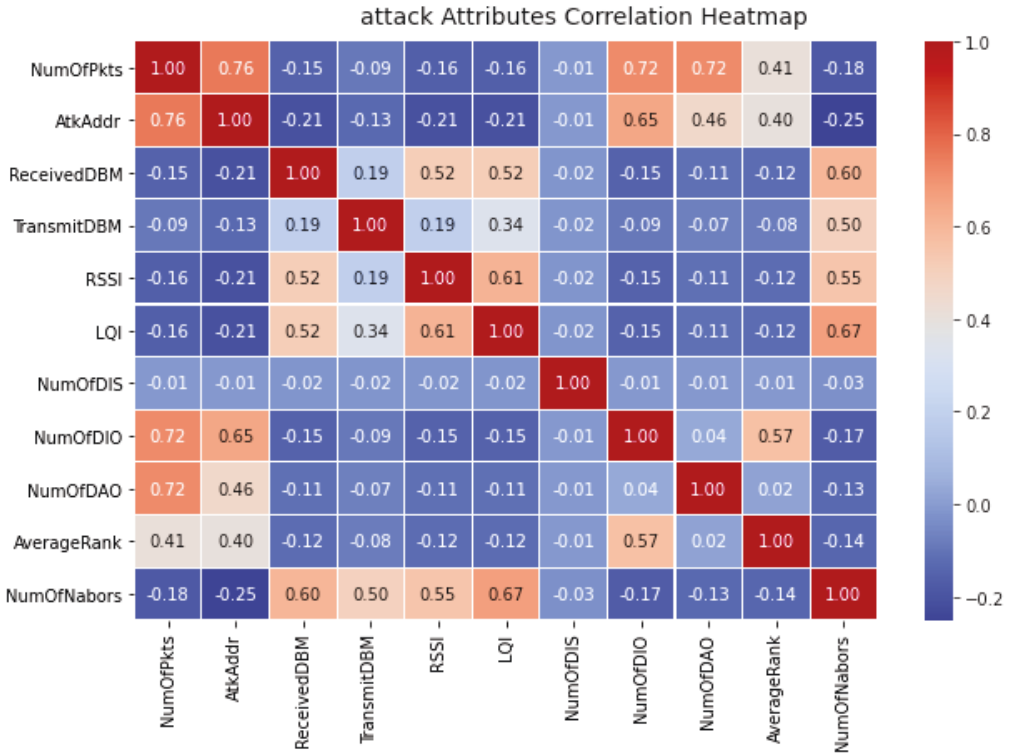


Figure 7.6 : IoT-DDoS heatmap

For further details about the collected IoT-DDoS refer to Chapter 6 of this thesis. In the next subsection, the data preparation and evaluation metrics are discussed. By examining different dimensions of the datasets and all the features relations detailed in Figure 7.6 as a heatmap, it can be seen that there is a significant

correlation between the number of DIS messages and a flooding attack. The number of DIS messages overwhelm the network to attack its neighbors. In machine learning terms, we correlate the relationship between all of the combined parameters and the result of detecting the attacks.

Training

In the training phase, the main task to be carried out is to fit the training dataset to the machine learning model. This is the initial step before validating and testing the model. However, before training the model with the dataset, the normalization and standardization process is carried out. Normalization is the process of transferring the dataset values to a normal range between 0 to 100, in our case. We use normalization since we have a large variance in feature values. In Python, we utilize the Sklearn library standard scaler, which subtracts the means (u) of each sample (x) and divides it by the standard deviation (s) of the whole dataset $z = (x - u)/s$ [122]. The training datasets consist of 50% of the no attack data, and the other 50% is divided equally and proportionally between the three attacks.

To reduce the dimensionality of the dataset, we use the principal component analysis(PCA) algorithm [54]. This allows us to obtain better results when training the model. In the next section, the validation and testing methods are discussed.

Validation and Testing

This phase's primary role is first to evaluate the result of the model and to tune the hyperparameters of the algorithm used. The validation step is used to ensure that the testing data is not used to evaluate the final result of the model. To avoid any bias in the result due to the test sub-datasets, the cross-validation technique is used. A specific subset of the dataset is shown in Figure7.4. As previously discussed, the three valid/test subsets of the datasets are used to validate and test the algorithms and to evaluate the accuracy of the result after the tuning process is done. The process at the phase is carried out recursively until the best model is identified. The cross-validation and hyperparameter tuning are explained as follows:

- **Cross-Validation:** is used in this study to avoid any problem with biased results in an unbalanced dataset. We use k-fold algorithm [122] to cross-validate the datasets. In its basic form, the k-fold cross validation creates a number of subsets of the training and testing datasets and alternates between the fold to ensure maximum real representation of the dataset. We use 6 folds for the k-fold algorithm. This means the datasets will be divided into six-folds to ensure unbiased results in training/testing.

- **Tuning the Hyperparameters:** The gridsearch framework [122] is used to search for the optimum parameters for the model. In grid-search, we define a set of values for the hyperparameters for each machine learning algorithm, and the grid-search automates the process of iteratively trying each hyperparameter for each model. This is done in sync with the cross-validation step to ensure a consistent result. To pipeline between the two processes, we use the pipeline library in Python.

A sample iteration of the grid-search used for cross validation and hyperparameters tuning for the SVM algorithm can be seen in Listing 1 below:

Metrics

The detailed metrics used in the thesis are explained in Chapter 5. Nonetheless, in this section, we highlight how the main metrics are used and how each parameter affects the end result:

Table 7.2 : Main matrices used

True Positive Rate	$\frac{TP}{Tp+FN}$
False Positive Rate	$\frac{FP}{FP+TN}$
F1-Score	$\frac{TP}{Tp+(FN+FP)/2}$

Precision: when we give a high value for precision we are telling the model to only mark the data point as being attack only if the model is completely confident

```
Grid scores on development set:

0.741 (+/-0.191) for {'SupVM__C': 0.1, 'SupVM__gamma': 0.006, 'pca': 2}
0.761 (+/-0.188) for {'SupVM__C': 0.1, 'SupVM__gamma': 0.007, 'pca': 2}
0.820 (+/-0.170) for {'SupVM__C': 0.1, 'SupVM__gamma': 0.008, 'pca': 2}
0.829 (+/-0.157) for {'SupVM__C': 0.1, 'SupVM__gamma': 0.009, 'pca': 2}
0.829 (+/-0.157) for {'SupVM__C': 0.1, 'SupVM__gamma': 0.01, 'pca': 2}
0.939 (+/-0.030) for {'SupVM__C': 0.1, 'SupVM__gamma': 0.02, 'pca': 2}
0.939 (+/-0.032) for {'SupVM__C': 0.1, 'SupVM__gamma': 0.03, 'pca': 2}
0.939 (+/-0.032) for {'SupVM__C': 0.4, 'SupVM__gamma': 0.006, 'pca': 2}
0.941 (+/-0.032) for {'SupVM__C': 0.4, 'SupVM__gamma': 0.007, 'pca': 2}
0.926 (+/-0.062) for {'SupVM__C': 0.4, 'SupVM__gamma': 0.008, 'pca': 2}
0.925 (+/-0.062) for {'SupVM__C': 0.4, 'SupVM__gamma': 0.009, 'pca': 2}
0.905 (+/-0.054) for {'SupVM__C': 0.4, 'SupVM__gamma': 0.01, 'pca': 2}
0.882 (+/-0.018) for {'SupVM__C': 0.4, 'SupVM__gamma': 0.02, 'pca': 2}
0.887 (+/-0.021) for {'SupVM__C': 0.4, 'SupVM__gamma': 0.03, 'pca': 2}
0.881 (+/-0.019) for {'SupVM__C': 0.9, 'SupVM__gamma': 0.006, 'pca': 2}
0.880 (+/-0.018) for {'SupVM__C': 0.9, 'SupVM__gamma': 0.007, 'pca': 2}
0.881 (+/-0.020) for {'SupVM__C': 0.9, 'SupVM__gamma': 0.008, 'pca': 2}
0.881 (+/-0.020) for {'SupVM__C': 0.9, 'SupVM__gamma': 0.009, 'pca': 2}
0.882 (+/-0.019) for {'SupVM__C': 0.9, 'SupVM__gamma': 0.01, 'pca': 2}
0.896 (+/-0.025) for {'SupVM__C': 0.9, 'SupVM__gamma': 0.02, 'pca': 2}
0.918 (+/-0.014) for {'SupVM__C': 0.9, 'SupVM__gamma': 0.03, 'pca': 2}
```

Listing 1: Grid search iteration

that it is an attack. This might cause a real attack to be classified as regular traffic, leading to high false negatives which means a lower recall value.

Recall: Sometime referred to as sensitivity, where it identify how sensitive it is to label normal traffic as malicious traffic. Causing a low precision result but will avoid the false negatives. It useful to have high recall value when we want to detect all attacks even if it means labeling some traffic as malicious.

If we have a specific goal in mind where we either want a higher precision and lower recall or vice versa, then we can simply use them as we see fit. However, if we do not have a clear goal, then we can simply use the F1 score which is the harmonic mean of the two values (precision and recall). In addition, the higher the number, the better the result of the machine learning model. Also, by having two parameters to judge how the algorithm performs, it is difficult to evaluate the algorithm. Therefore, we use the F1 score as it is a combined metric of the two to evaluate the model. Nevertheless, we list precision and recall as a reference. Also,

the ROC curve will be drawn to show the value of the F1 measure. Moreover, since we are dealing with attacks, we want to test different recall values to see how the method performs, we have chosen different recall values since it is acceptable as a preventative measure to mark some of the regular packets as an attack but make sure to detect the attacked nodes. Table 7.7 shows how we calculated the true positive rate, the false positive rate, and the F1-score, which are crucial for the attack detection results.

7.4 Machine Learning Detection Evaluation

In this section, the three methods are discussed individually, and at the end of this section, a comprehensive comparison between them is presented. Furthermore, the full implementation of the thesis' experiments including figures and the detailed results of each method can be found in a GitHub repository [123]. Here, we only present the end result and the best model generated after the iterative validation and testing process:

7.4.1 SVM Experiment

This experiment uses the Sklearn [122] Python library to implement the SVM algorithm. As previously mentioned, the RBF kernel is used since the data are not linearly separable.

Table 7.3 : SVM confusion matrix comparison

Label	Blackhole	SF	Flood
TP	97.8%	99.0%	99.0%
TN	92.8%	73.0%	100.0%
FP	0.0%	27.0%	0.0%
FN	0.0%	0.8%	0.8%

The hyperparameters of the SVM were discussed in the previous section. Figure 7.7 shows that the SVM confusion matrix achieves good detection results for blackhole and flooding attacks. However, there is a slight decrease in accuracy result

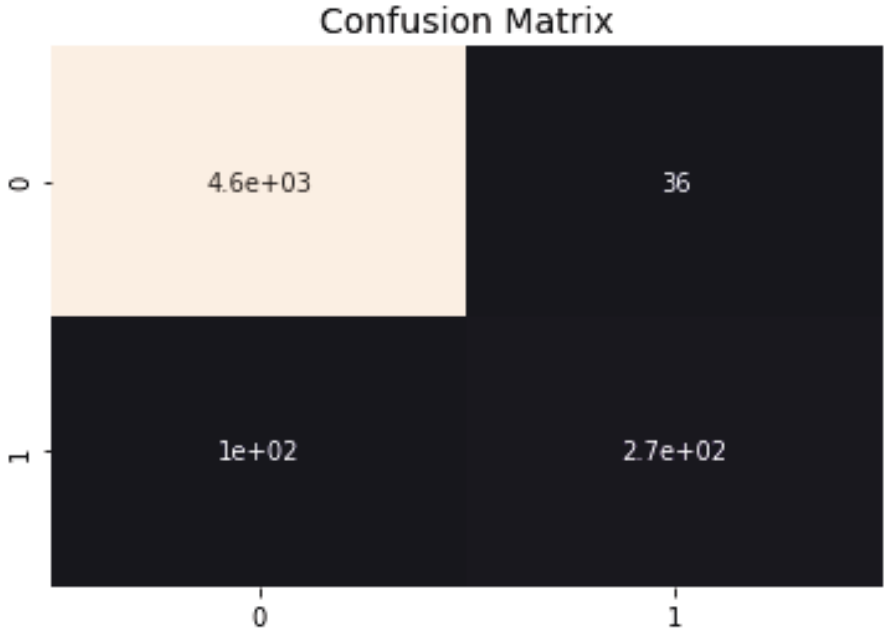
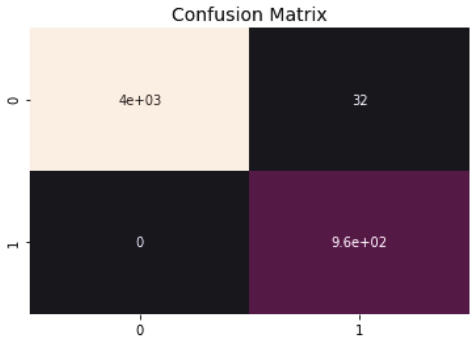
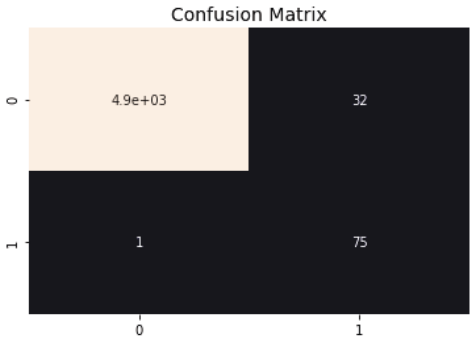


Figure 7.7 : SVM confusion matrix BH



(a) SVM confusion matrix flood



(b) SVM confusion matrix SF

in relation to the selective forwarding attack. As can be seen from Table 7.3, the true negatives(TN) of SF decrease compared to the other attacks. This is because forwarding attacks generate a large amount of fake traffic that looks identical to normal traffic.

Figure 7.9 shows the best performance of the SVM method in detecting all the attacks. Table 7.4 shows the result of one of the many experiments conducted to tune the SVM model. Therefore, after extensive and rigorous testing and evaluation, we conclude that the best parameters for the SVM model are as follows:

- C: This represents the smoothness of the hyperplane when classifying the

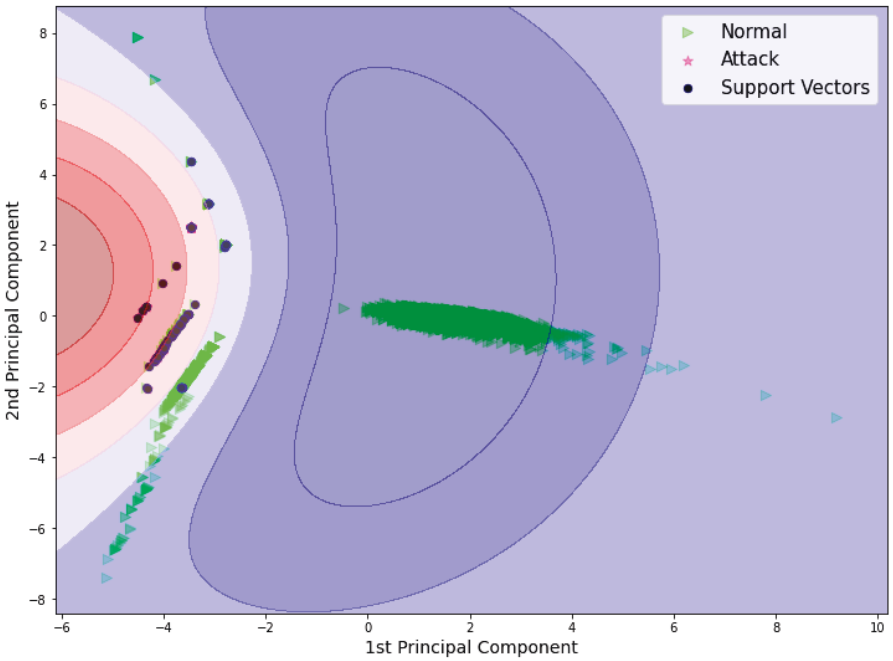
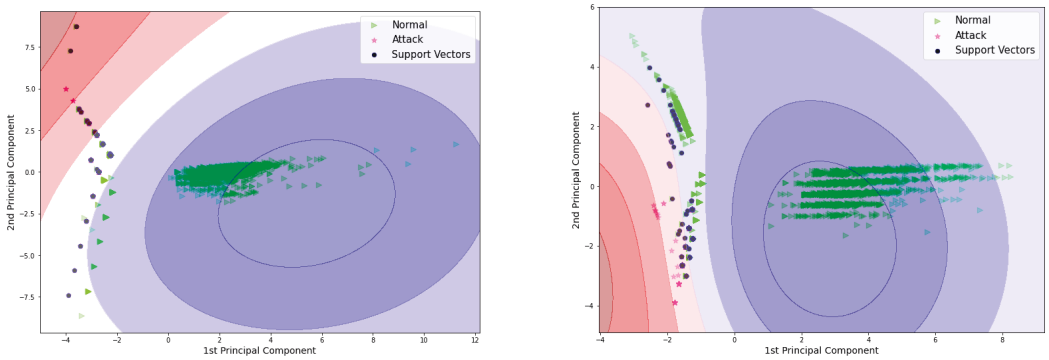


Figure 7.9 : SVM hyperplane separation SF



(a) SVM confusion matrix BH

(b) SVM confusion matrix flood

points. A large value of C means more points will be classified correctly. The C variable is part of the soft margining in the slack variable equation discussed earlier in section 7.2. For our example, we make $C = 2$

- Gamma: This is a variable that exists only on the RBF kernel and it is used to determine the influence of a single data point on the curvature of the decision boundary. We use $gamma = 0.00.1$, to obtain the best model result from the SVM algorithm.

Figure 7.9 shows the SVM separation hyperplane of the best model selected for

Table 7.4 : Example of one of the K-fold tests

Result f1	Gamma	C
0.492 (+/-0.000)	0.1	0.006
0.492 (+/-0.000)	0.1	0.007
0.492 (+/-0.000)	0.1	0.008
0.492 (+/-0.000)	0.1	0.009
0.492 (+/-0.000)	0.1	0.01
0.492 (+/-0.000)	0.1	0.02
0.492 (+/-0.000)	0.1	0.03
0.492 (+/-0.000)	0.4	0.006
0.492 (+/-0.000)	0.4	0.007
0.492 (+/-0.000)	0.4	0.008
0.492 (+/-0.000)	0.4	0.009
0.492 (+/-0.000)	0.4	0.01
0.524 (+/-0.108)	0.4	0.02
0.899 (+/-0.325)	0.4	0.03
0.492 (+/-0.000)	0.9	0.006
0.492 (+/-0.000)	0.9	0.007
0.492 (+/-0.000)	0.9	0.008
0.534 (+/-0.144)	0.9	0.009
0.524 (+/-0.108)	0.9	0.01
0.677 (+/-0.212)	0.9	0.02
0.756 (+/-0.271)	0.9	0.03

the three attacks. Overall, all of the hyperplanes were able to separate the attacks efficiently. Only in the case of SF attacks was there a slight decrease in the number of the attacks false labeled as positive.

7.4.2 ANN Experiment

For neural networks, we have used the MLP classifier with backpropagation with the following settings:

- Hidden layer: This is used to specify how many layers there are between the input of the network and the output of the network and how many neurons there are in each hidden layer. The higher the number, the more time it takes to process, but the more accurate the results. In this example, we use the value (5,4).

- Alpha: used for regularization in other terms, it is the penalty value used to specify the size of the weights used against overfitting. In some cases, the high

alpha value is used to fix the overfitting problem where there is high variance. On the other hand, it can also be used to fix the problem of underfitting by lowering the alpha value. In our case, the best alpha value we found after many testing runs is 0.001. We used the grid-search framework to find this best result.

Table 7.5 shows the precision, F1, and recall of the algorithm. As can be seen, the ANN does not perform well for our datasets, especially for flooding attacks. This can be explained due to the large amount of DIS messages sent, which might cause the algorithm to perform poorly.

Furthermore, the confusion matrix in Figure 7.11 shows a high number of false positives, and the number of outliers detected was nearly 0.

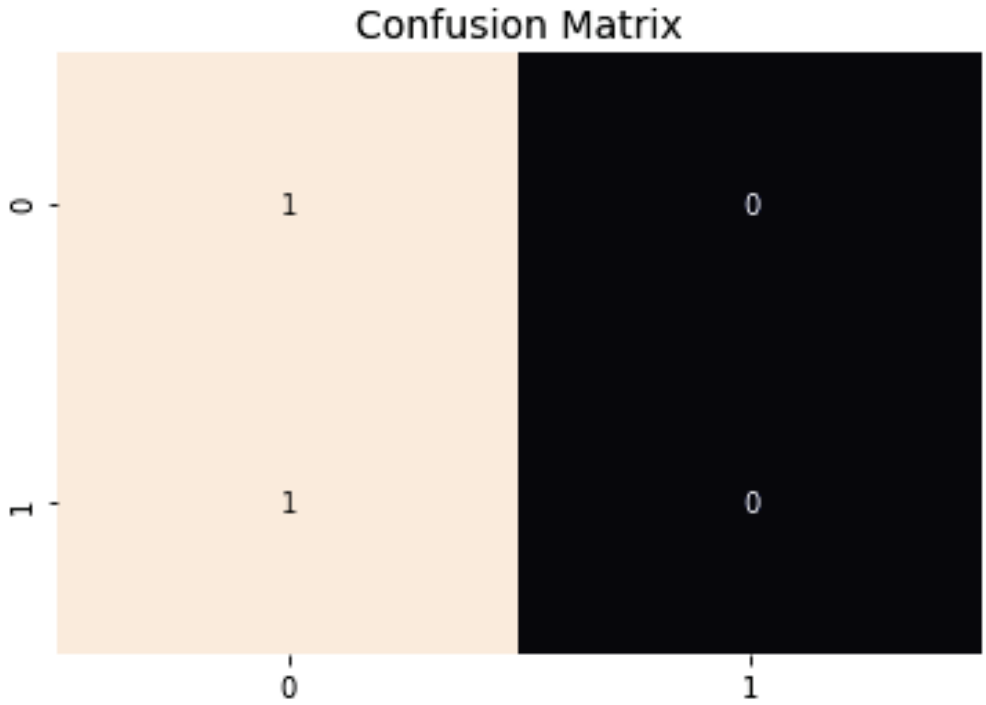


Figure 7.11 : Confusion matrix for artificial neural network

An examination of Table 7.5 shows that the ANN algorithm performed poorly compared to the SVM. Although we mainly focus on the logistic activation function for comparison, we include the ReLU activation method. The ReLU method

Table 7.5 : Example of one of the K-fold tests

Result F1 Average	Activation	Alpha	Hidden layer	Learning rate
0.745 (+/-0.507)	logistic	0.0001	(1, 20)	constant
0.759 (+/-0.456)	logistic	0.0001	(1, 20)	adaptive
0.796 (+/-0.498)	logistic	0.0001	(50, 50, 50)	constant
0.745 (+/-0.507)	logistic	0.0001	(50, 50, 50)	adaptive
0.759 (+/-0.456)	logistic	0.0001	(50, 100, 50)	constant
0.745 (+/-0.507)	logistic	0.0001	(50, 100, 50)	adaptive
0.730 (+/-0.253)	logistic	0.0001	(100,)	constant
0.692 (+/-0.491)	logistic	0.0001	(100,)	adaptive
0.692 (+/-0.491)	logistic	0.05	(1, 20)	constant
0.796 (+/-0.498)	logistic	0.05	(1, 20)	adaptive
0.745 (+/-0.507)	logistic	0.05	(50, 50, 50)	constant
0.798 (+/-0.004)	logistic	0.05	(50, 50, 50)	adaptive
0.745 (+/-0.507)	logistic	0.05	(50, 100, 50)	constant
0.796 (+/-0.498)	logistic	0.05	(50, 100, 50)	adaptive
0.759 (+/-0.456)	logistic	0.05	(100,)	constant
0.759 (+/-0.456)	logistic	0.05	(100,)	adaptive
0.730 (+/-0.253)	relu	0.0001	(1, 20)	constant
0.730 (+/-0.253)	relu	0.0001	(1, 20)	adaptive
0.692 (+/-0.491)	relu	0.0001	(50, 50, 50)	constant
0.692 (+/-0.491)	relu	0.0001	(50, 50, 50)	adaptive
0.573 (+/-0.440)	relu	0.0001	(50, 100, 50)	constant
0.553 (+/-0.440)	relu	0.0001	(50, 100, 50)	adaptive
0.692 (+/-0.491)	relu	0.0001	(100,)	constant
0.759 (+/-0.456)	relu	0.0001	(100,)	adaptive

performed poorly compared to the logistic method. Table 7.5 provides a sample of one of the cross-validations with which we experimented. This example is by far the best example we have seen. Although we can see when the hidden layer number increases, there is a very slight improvement. However, it is not to the level where it outperforms the SVM method. Furthermore, if we look into the confusion matrix shown in Figure 7.11, we can see that the result is biased, and none of the attack samples have been identified as attacks, which indicates that the algorithm cannot be trusted to detect attack samples of the dataset. The performance of the MLP decreased dramatically when we use all the attack dataset, recording a low average

F1 score of 0.2 compared to 0.8 for the SVM algorithm.

7.4.3 Random Forest Experiment

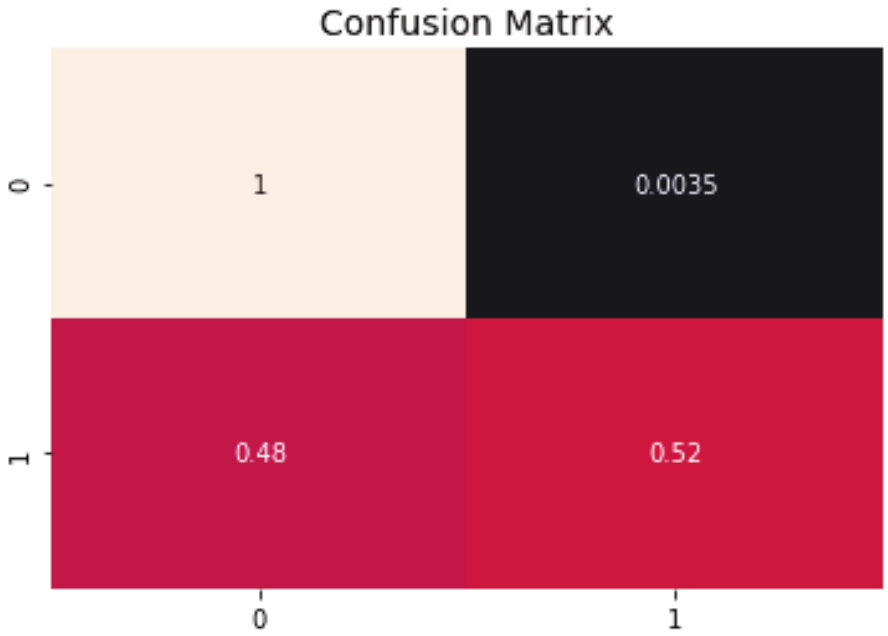
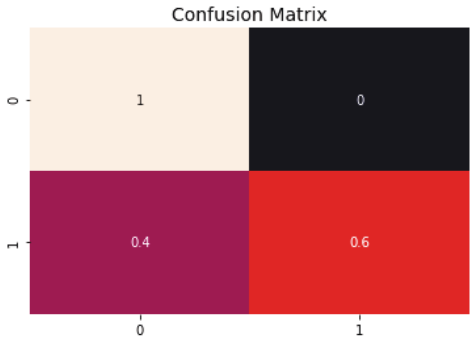
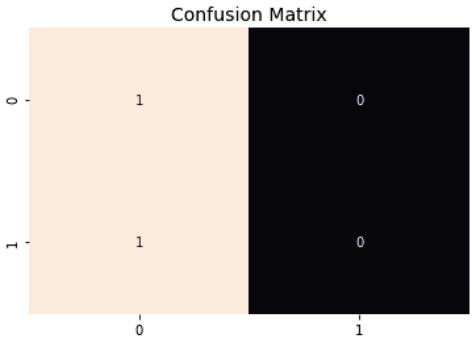


Figure 7.12 : Random forest confusion matrix for blackhole attack



(a) Random Forest confusion matrix for flood attack



(b) Random forest confusion matrix for SF attack

As explained in this chapter in section 7.2, the random forest is a special type of decision tree widely used for classification and regression. The IoT-DDoS is used as an input and the Sklearn random forest classifier. Before we detail the results, there are different hyperparameters involved in tuning the random forest classifier, each of which is explained as follows:

- The N estimator: is a hyperparameter used to define the number of trees produced using the classifier. A random forest with n estimator = 5 means that the classifier will produce five decision trees.
- Criterion: defines which splitting method is used to split the decision tree into leaves and calculate the result. Two common criterion types are used in this experiment the Gini impunity and the entropy method.
- Max depth: specifies the depth of the tree downward. The higher the number, the more complex the tree, leading to a longer processing time.

Table 7.6 : RF hyperparameter tuning

Result	Tree criterion	max depth	n_estimator
0.660 (+/-0.201)	gini	1	1
0.660 (+/-0.201)	gini	1	1
0.493 (+/-0.000)	gini	1	2
0.871 (+/-0.436)	gini	1	3
0.744 (+/-0.503)	gini	1	4
0.493 (+/-0.000)	gini	1	5
0.493 (+/-0.000)	gini	1	6
0.660 (+/-0.201)	gini	1	1
0.660 (+/-0.201)	gini	1	1
0.493 (+/-0.000)	gini	1	2
0.871 (+/-0.436)	gini	1	3
0.744 (+/-0.503)	gini	1	4
0.493 (+/-0.000)	gini	1	5
0.493 (+/-0.000)	gini	1	6
0.539 (+/-0.159)	entropy	1	1
0.539 (+/-0.159)	entropy	1	1
0.493 (+/-0.000)	entropy	1	2
0.618 (+/-0.435)	entropy	1	3
0.493 (+/-0.000)	entropy	1	4
0.493 (+/-0.000)	entropy	1	5
0.493 (+/-0.000)	entropy	1	6
0.539 (+/-0.159)	entropy	1	1
0.539 (+/-0.159)	entropy	1	1
0.493 (+/-0.000)	entropy	1	2
0.618 (+/-0.435)	entropy	1	3
0.493 (+/-0.000)	entropy	1	4
0.493 (+/-0.000)	entropy	1	5
0.493 (+/-0.000)	entropy	1	6

Figure 7.12 shows the confusion matrix of the random forest model applied to the three attack datasets. As can be seen, the result were poor compared to the ANN model and SVM model. Although the RF model was able to detect the flooding attack with a good result, the other two attacks generated high false-positives, hence they cannot be used as reliable detection models for the IoT-DDoS dataset. Furthermore, the selective forwarding attack suffered the most from the 0 detection

of true negatives (TN), representing the attack data points. This indicates that selective forwarding attacks are very difficult to detect using the RF algorithm. Table 7.6 shows examples of one of the many experiments that we conducted using cross-validation and grid-search to tune the RF model. A detailed comparison between each of these models is presented in the next section.

7.5 Discussion and Evaluation

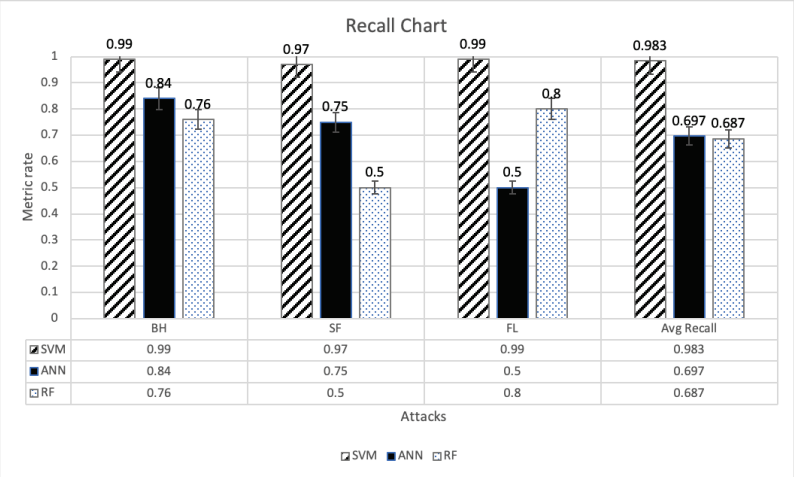


Figure 7.14 : Recall chart comparing SVM, ANN and RF

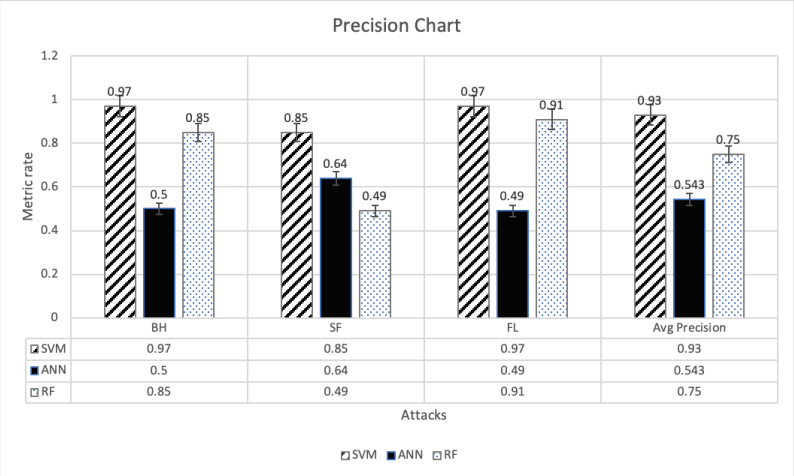


Figure 7.15 : Precision chart comparing SVM, ANN and RF

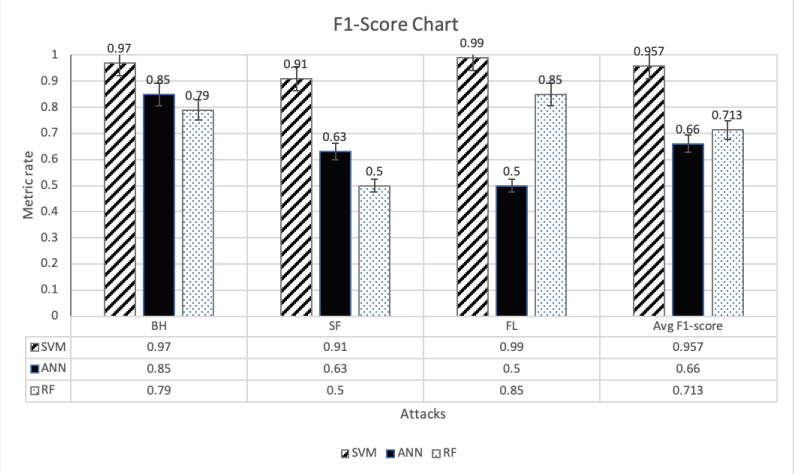


Figure 7.16 : F1-score chart comparing SVM, ANN and RF

In this section, we discuss the results of all the algorithms. We run more than 80 experiments with different parameters and tuning to obtain an acceptable result for each method, as can be seen from Table 7.7.

The total number of experiments per attack depends on the number of parameters to be tuned and the number of variances. To elaborate, let us take the SVM method as an example if the number of variables to be changed is 2 (C and gamma), and the number of variances equals 3 and 7 respectively and multiplying these two values gives a 21 classifier fit per fold. If we assume that we have four k-fold cross-validations, the number of experiments equals $4 * 21 = 84$. Although this tuning process is time-consuming, it allows an accurate selection of the model hyperparameters. The results show that the SVM model is the best performer overall compared to all of the methods used. Table 7.7 shows the results of all of the methods. As can be seen, the SVM algorithm performs the best in all aspects of attack detection, as shown in the table. Although it achieves results which are 10% lower than all the others for the flooding attack, it performed 30% better than ANN and 50% better than RF for the same attack. In Figure 7.14, 7.15, and 7.16 we can see the precision, recall and the F1-score for each method and for each attack. The SVM algorithm outperforms all of the other detection methods in terms of the number of attacks detected. The performance of these two approaches is observed based

on accuracy, false negative rate, and precision. The results indicate that the SVM classification ability produces more accurate results than RF and RF takes less time to train the classifier than SVM. Research in intrusion detection using the SVM and RF approach remains a popular topic, due to their good performance. The findings of this thesis will be beneficial for future research and will assist SVM and RF to be used in a more meaningful way to maximize performance and minimize the false negative rate.

Table 7.7 : Final results

Classifier	BH	SF	FL	Avg
SVM	0.970	0.910	0.990	0.957
ANN	0.850	0.630	0.500	0.660
RF	0.790	0.500	0.850	0.713

(a) F1-Score

Classifier	BH	SF	FL	Avg
SVM	0.990	0.970	0.990	0.983
ANN	0.840	0.750	0.500	0.697
RF	0.760	0.500	0.800	0.687

(b) Recall

Classifier	BH	SF	FL	Avg
SVM	0.970	0.850	0.970	0.930
ANN	0.500	0.640	0.490	0.543
RF	0.850	0.490	0.910	0.750

(c) Precision

7.6 Conclusion

In this chapter, we presented a machine learning approach to analyze and intelligently detect three kinds of malicious attacks in IoT. SVM, neural network, and random forest are the methods discussed in this chapter, showing promising results when combined with the dataset generated in Chapter 6 of this thesis. Furthermore, we also presented a comprehensive SVM model tuned to detect DDoS attacks in IoT. We also comprehensively analyzed the performance of each of the machine learning methods to choose the best model to be used in the next chapter. In the next chapter of this thesis, we implement the best model from this chapter into our complete IDS solutions, alongside the real-time data gathering framework discussed in Chapter 6.

Chapter 8

A Machine Learning IDS Implementation in IoT Networks

8.1 Introduction

This chapter presents the intrusion detection system implementation and all the associated algorithms. IoT-DDoS aims at detecting anomalous nodes and isolates them from the network. To achieve this, we implement a real-time monitoring unit that sends malicious node traffic to the IDS unit installed in the 6BR router. The IoT-DDoS [124] design takes into account all of the constraint factors associated with IoT devices and tries to detect attacks without compromising either data integrity or node performance. Using the machine learning model implemented in Chapter 7, we embed it into our IDS solution and use it as an attack classifier for the network. The proposed system consists of three components, real-time data acquisition, detection unit and the IDS agent, which is discussed in section 8.2. In section 8.3, we describe the implementation of the solution and embed the machine learning model in the 6BR router and the deployment of ML is discussed. In section 8.4 the IDS evaluation and results are discussed. To give an example as to how the IDS can be utilized, in section 8.5 an example of web application is presented. Finally this chapter is concluded in section 8.6

8.2 IDS Implementation in the IoT Network

In this section, we present a novel hybrid IDS implementation for detecting DDoS attacks in IoT networks. Before this, we explore the different components of the IDS shown in Figure 8.1, which is a detailed framework of the real-time detection system and the placement of the different parts of the IDS. This framework is not different from the framework introduced in Chapter 5; rather, it complements it by exploring

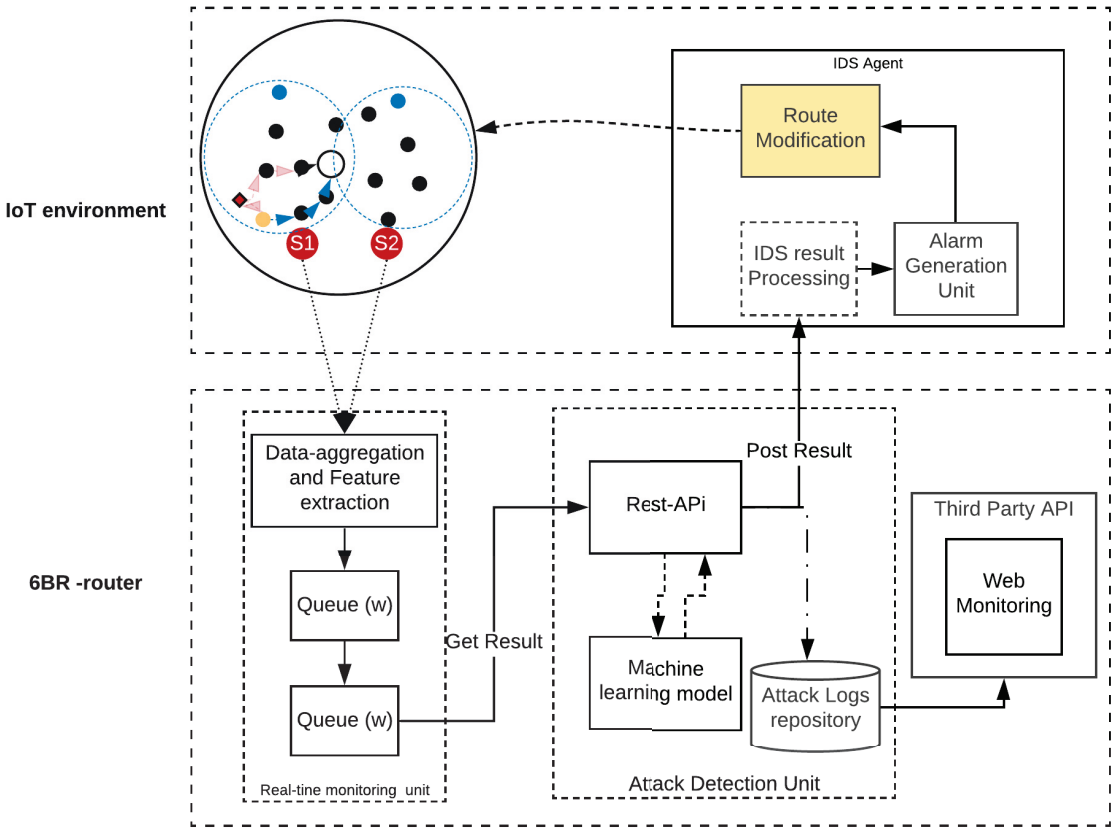


Figure 8.1 : Overview of the IDS framework

all of the components involved in the process. We divide the framework into two main layers, the IoT network layer and the 6LoWPAN border router, as shown in Figure 8.1. This division will allow us to understand the entire life cycle of the attack detection mechanism from when the attack is initiated until the IDS has detected it. We explore the different units involved in the IDS operations, which are organized sequentially as follows:

8.2.1 Real-Time Data Monitoring and Aggregation Unit

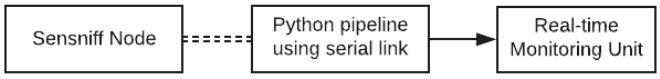


Figure 8.2 : Pipeline for the real-time data monitoring and aggregation unit

This unit and its implementation was discussed and in Chapter 6; the only modification introduced in this chapter is the ability to perform the process in real-time instead of reading the data from a PCAP file, hence we have designed an online parser that can read the data and parse them in real-time. We achieve this by using the Sensniff sniffing node and create a pipeline using Python and passing it to the real-time monitoring unit. The link between the Python pipeline script and the IoT sniffer node is achieved through a serial link. The python code used to link the sniffer node and the IDS monitoring unit is as follows:

```
import serial
con = serial.Serial(
   port='/dev/ttyUSB0',\
   baudrate=115200,\
   bytesize=serial.EIGHTBITS,\
   timeout=0)
print("connected to: " + con.portstr)
con.write("help\n");
while True:
    data = con.readline();
    if data:
        print(data),
con.close()
```

This code is responsible for interfacing the sniffers with our real-time monitoring unit. The rest of the monitoring unit has already been explained in detail in Chapter 6.

8.2.2 Attack Detection Unit

This unit is where most of the core work is done. It is responsible for using the machine model trained and tested in Chapter 7 to detect and classify malicious node based on the features extracted from the Real-Time Feature Extraction Unit. To

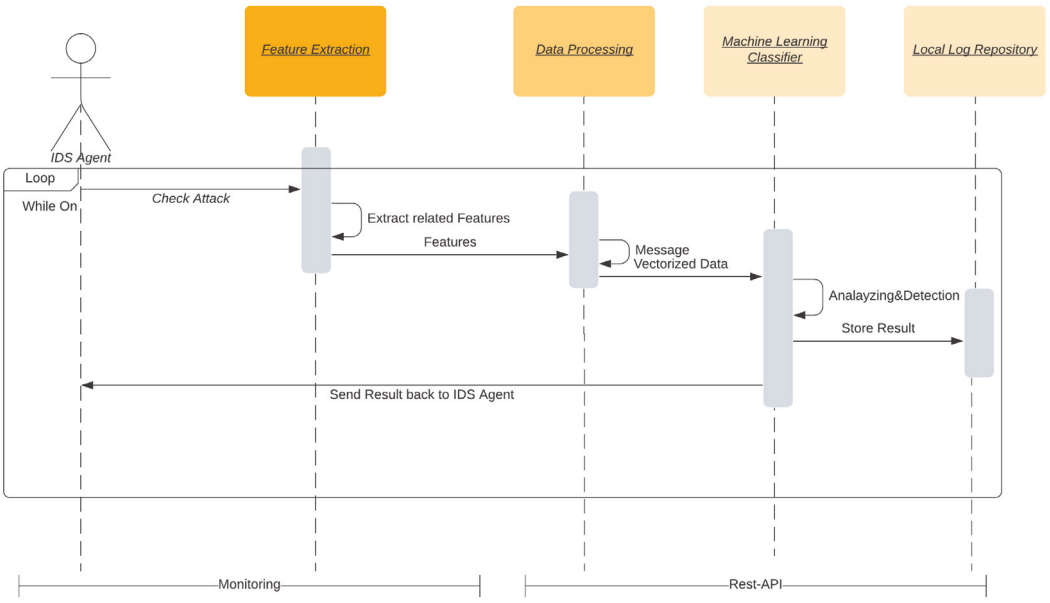


Figure 8.3 : Sequence diagram for the IDS framework

implement this unit, we have used the Sklearn [122] framework to build the model and the pickle library to export the trained model. Figure 8.3 shows the sequence diagram of the whole machine learning process.

REST-API

The REST-API is part of the detection unit and it handles getting and posting of the results to the IDS agent. Figure 8.3 shows the steps involved in handling all of the API operations in reaction to the whole IDS. In section 8.2 of this chapter, we explained the REST-API and its implementation in detail.

Algorithm to Notify IDS Agent About the Attacker Node

If the machine learning model detects an outlier, an alarm is sent to the IDS agent through the serial link that has been previously explained. The primary purpose of this message is to notify the IDS agent about the attack and its location. Figure 8.4 it shows the structure of the packet sent to the IDS agent. This packet contains various parameters, the attacker node ID, the node location in relation to the RPL network and the result of the attack. It is essential to specify the type of attack

since the IDS agent behaves differently for each attack. Following is a code snippet from the main part of the serial link handler.

```
#This code will write to the serial link
#the result of the machine learning model
#to the IDS agent.

ser.write(Alert);
while True:
    line = ser.readline();
    if line:
        print(line),
ser.close()
```

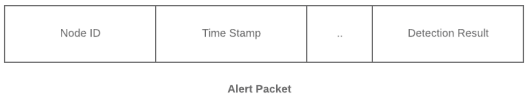


Figure 8.4 : Alert packet

This code connects the IDS router to the IDS agent through the serial link and sends the result to the serial link. This handler takes care of decoding the message and broadcasting an alert message to the affected nodes. The structure of the IDS Alert message can be seen in Figure 8.4

8.2.3 IDS Agent Implementation

The IDS agent works as a sink node to where all the nodes in the network send the data. Moreover, since our network is designed to send information using the UDP protocol, the sink node works as the UDP server, where it handles all of the UDP packets coming from each node. The UDP packets coming from the clients contain sensor information such as temperature and humidity. However, the sink node also is responsible for handling alert messages sent from the IDS unit. The IDS agent's main functions are summarized in the following subsections.

Algorithm to Isolate Attacker Node From the Network

Algorithm 8: Notify algorithm

```
Blacklist     : Nodes= {N_1,...,N_i} of size i
Packet(Alert packet with all of its input)
Attackers ← list of all attackers in each Blacklist of size i;
s = Attackers.size ;
for j ← 0 to s do
    if Attackers [j].size != null then
        add the Packet(Attackers.ip) to the Blacklist
        SendBodcast(Attackers.ip);
        if Packet(Attackers.ip).type = (SF or Blackhole) then
            fixToplogy();
        else  dontFixToplogy. ;
    else Do nothing the attacker already blocked;
```

To isolate the attacker node from the rest of the network, we developed algorithm 8 which maintains a blacklist of all of the malicious nodes identified by the IDS. In addition to maintaining a blacklist, this algorithm handles the process of sending the broadcast and initiating DoDAG Reconstruction when blackhole and selective forwarding attacks are detected. These attacks affect the topology of the network by manipulating node ranking, therefore altering the structure of the DODAG tree.

The *sendbodcats*() method used in the notify algorithm is implemented as part of the IDS agent methods. Furthermore, since the IDS agent runs the Contiki Operating system, we utilize the *uip_create_linklocal_allnopdes_mcast*() method which is part of the network architecture of the OS. The following code is a small part of the broadcast method utilized.

```
void sendbodcast(addr) {
 printf("Sending broadcast\n");
 uip_create_linklocal_allnodes_mcast(&addr);
 simple_udp_sendto(&broadcast_connection, "Test", 4, &addr);
}
```

Algorithm to Fix network Topology After an Attack

As previously mentioned, algorithm 8 s responsible for checking the type of attack to initiate topology reconstruction. Fortunately, the process of fixing the topology in Contiki OS [125] is already implemented; so we only have to call the method when needed. However, we add a few changes to ensure a consistent DAG version across the whole network. The following code keeps track of the number of network fixes that were called and also triggers the *global_repair* method which is part of the network core module of Contiki OS. Furthermore, *rpl_rest_dio_timer*() method is called to rest the DIO timer, which is crucial to ensure RPL consistency.

```
if(lollipop_greater_than(dio->version, dag->version)) {
    if(dag->rank == ROOT_RANK(instance)) {
      dag->version = dio->version;
      RPL_LOLLIPOP_INCREMENT(dag->version);
      rpl_reset_dio_timer(instance);
    } else {
      global_repair(from, dag, dio);
    }
    return;
  }
```

8.3 Deployment of the Machine Learning Model

There are many machine learning deployment methods, and it is essential to choose the right one to avoid any resource complications. The most common method is using a data streaming pipeline method where we have a large number of data coming from different sources at the same time. This kind of method requires a very complicated system design to handle big data. The other common method is what is called train by batch and predict on the fly using REST-API frameworks. The concept is to ask for prediction on demand when there is a request applicable for a simple application that requires prediction only when needed. In this research, the most suitable approach is to use the streaming approach since we are dealing

with many data input IoT devices. However, the end goal of this research is to test how the designed machine learning algorithm perform in the IoT environment. We take a different approach instead of using the node streaming method and dealing with a different level of system components. We designed a simple framework that combines the use of our own built data-aggregator and then passes it to the REST-API machine learning server.

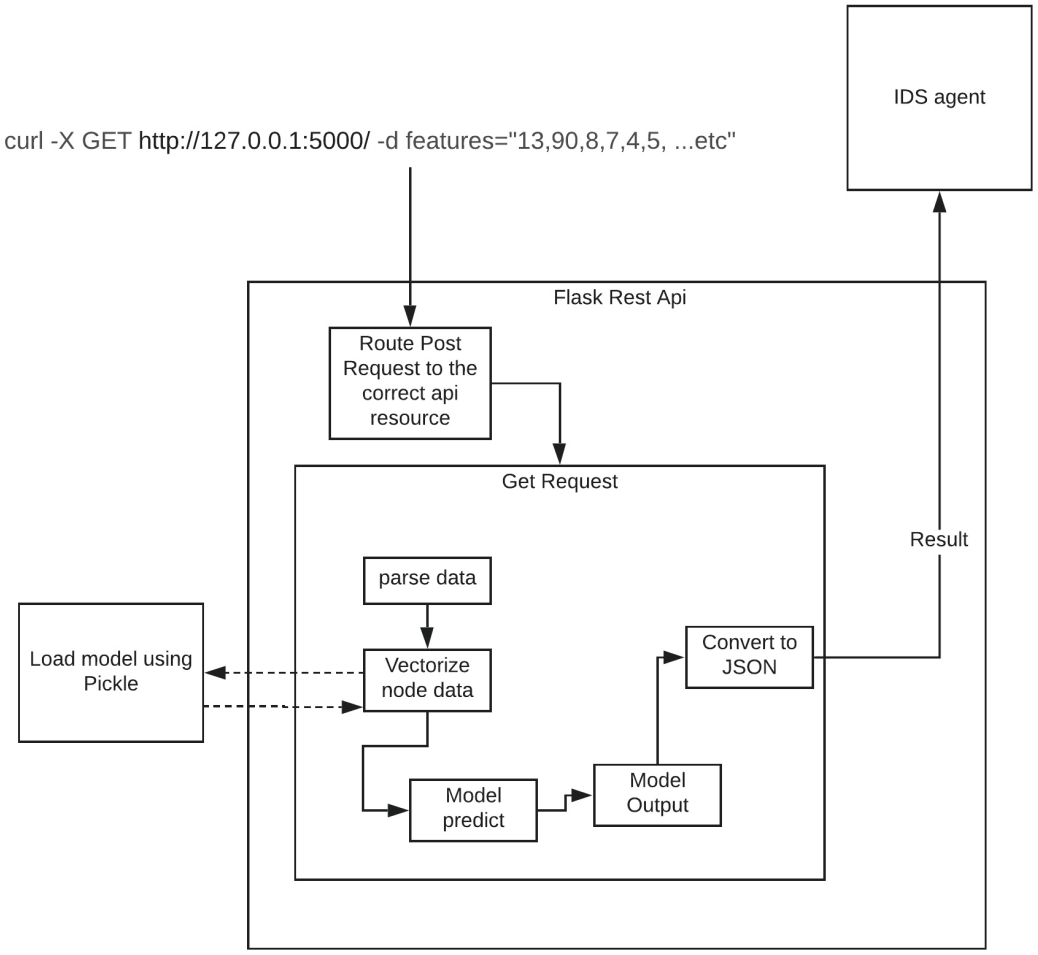


Figure 8.5 : The IDS framework REST-API unit used for third party plugins

The deployment of any machine learning model is a complicated process, and it must be carefully designed and tested. Therefore, we divide this process into multiple steps sequentially, each of which is explained as follows:

1. The first step is to extract the machine learning model that we discussed in Chapter 7. For this process, we make sure that we achieve a satisfying result in terms of the accuracy and the precision of the model (for further details, please refer to Chapter 7). To export the model, we use the pickle library which is available as part of Python 3.7. The following code snippets show the process of how the model was exported :

```
import pickle
pickle.dump(clf, open('models/SVMClassifer.pickle', 'wb'))
```

   This simple code exports the entire trained and tested model as a pickle file to be used at a later stage of this deployment. This step is implemented after all the training and testing of the model has been finalized, and we are happy with the result. We have already compared and evaluated three machine learning models, and the best performing model was the SVM model, as explained in Chapter 7.

2. The second step is to use the real-time aggregation and feature extraction unit to aggregate and extract the relevant features from the streamed code. To achieve this, we built an online extraction unit based on the offline extraction unit discussed in Chapter 6. This version considers the multiple streaming inputs coming from the sniffer nodes and passes them to the feature extraction unit. The only difference in the aggregation unit at this stage is that instead of exporting the data to a CSV file, we created a simple data parser that sends the data to the REST-API as a get request.

3. The third step is to build a REST-API stack; in Python, there are multiple frameworks that can be used to build a REST-API. For this project, we chose the flask framework [126] due to its popularity and robustness to build the API server. Following is a sample code of the get request used to obtain the result from the trained model, and the process is explained in the following steps sequentially:

Figure 8.5 elaborates how each of the components interacts with the REST-API. The whole process can be summarized as follows:

(a) The first step of the REST-API module is to take the get request from the extraction unit where the request is embedded with the node features to be analyzed into the data argument.

```
from flask import Flask
from flask_restful import reqparse, abort, Api, Resource
import pickle


app = Flask(__name__)
api = Api(app)
# Creating a SVM model object
model = SVC()
# Load tested and trained model
clf_path = 'models/SVMClassifer.pkl'
with open(clf_path, 'rb') as f:
    model.clf = pickle.load(f)
# load model vectorizer
vec_path = 'models/SVMVectorizer.pkl'
with open(vec_path, 'rb') as f:
    model.vectorizer = pickle.load(f)
```

(b) The second step involves the flask API re-routing the request to the correct resource. In our case, the resource class is called the *attackPredication* class .

(c) In the third step, the get method is called. It handles data parsing and vectorization along with parsing the vectorized data to the trained model and generating the output as JSON data.

```
class attackPredetion(Resource):
    def get(self):
        # use the realtime 6LoWPAN parser
```

```
#to get the data
args = parser.6LoWPANData()
# vectorize the user's query and make
#a prediction
vectorized = model.vectorizer_transform(
    np.array([args]))
prediction = model.predict(vectorized)
pred_proba = model.predict_proba(vectorized)


# convert result as text
#to be stored in local repository

if prediction == 0:
    pred_text = 'Normal'
else:
    pred_text = 'Attack'

confidence = round(pred_proba[0], 3)
#  JSON object to be transfered to IDS agent
output = {'prediction': pred_text,
'confidence': confidence, 'attackerIp': args['ip'}

return output
```

The get method takes the data argument(features) and parses the input from the get request using the **reqparse** library in Python. The parsed data is then prepared and vectorized using $model.vectorizer_transform$ method, and later on is passed to the trained machine learning model. As can be seen from the code above, the predict model generates the result as zeros or ones, therefore, we convert these outputs into either normal or attack. This output is later converted into JSON format to be passed

to the IDS agent through the serial link.

The next section of this Chapter evaluates the aforementioned system.

8.4 Evaluation

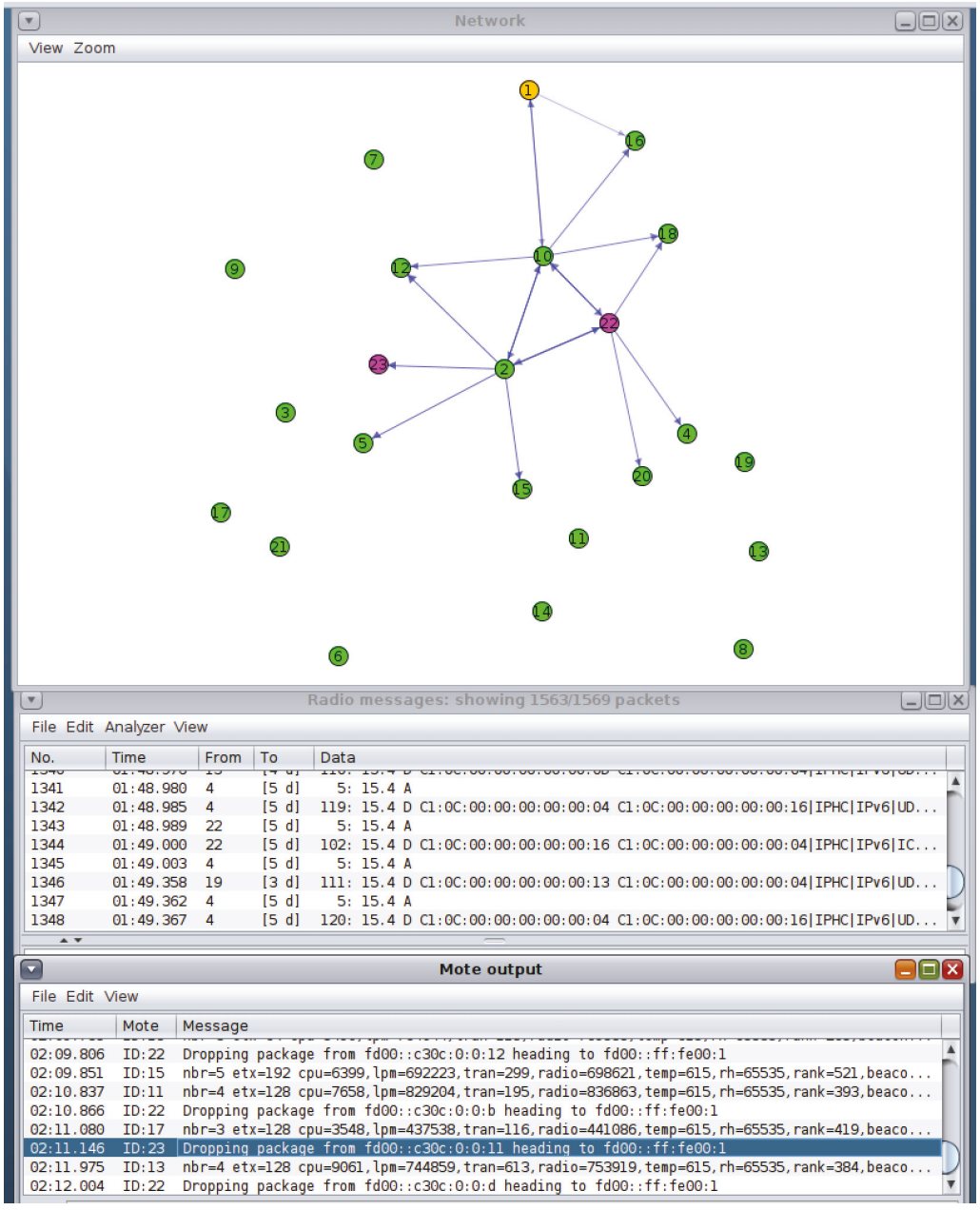


Figure 8.6 : Blackhole attack implementation

The evaluation process can be summarized into two parts, the evaluation metrics to evaluate the performance of the IDS and the IDs performance under different

scenarios. The main metrics used to evaluate any IDS is the true positive rate (TPR) and the false positive rate, which were explained in Chapter 4 of this thesis. They can be summarised as follows:

TPR = TP/TP+FN

FPR = FP/FP+TN

These metrics are used to evaluate the IDS performance under the followings scenarios explained as follows :

Table 8.1 : Detection rate table

Attack	TP	FP	TN	FN	TPR (%)	FPR(%)
SF	100	2	6412	0.5	99.50%	0.03%
BH	94	5	6291	0.3	99.68%	0.08%
Flood	87	8	5901	0.2	99.77%	0.14%

The first scenario is the accuracy of attack detection. When attacks are initiated at the beginning of the simulation, as can be seen from Table 8.2, our IDS shows promising results with a high detection accuracy of 99.50%. At this stage, the amount of network traffic is still minimal, and the high accuracy rate can be explained because the attacker node has not been given a chance to adapt to the network and deceive as many nodes as it can, especially for the blackhole and selective forwarding attack. Therefore, the IDS was able to blacklist and isolate the node immediately.

Table 8.2 : Detection rate table

Attack	TP	FP	TN	FN	TPR (%)	FPR(%)
SF	520	235	43231	7	98.67%	0.54%
BH	423	323	33231	13	97.02%	0.96%
Flood	334	376	53231	14	95.98%	0.70%

However, for the second scenario where the IDS operation is introduced 10 min-

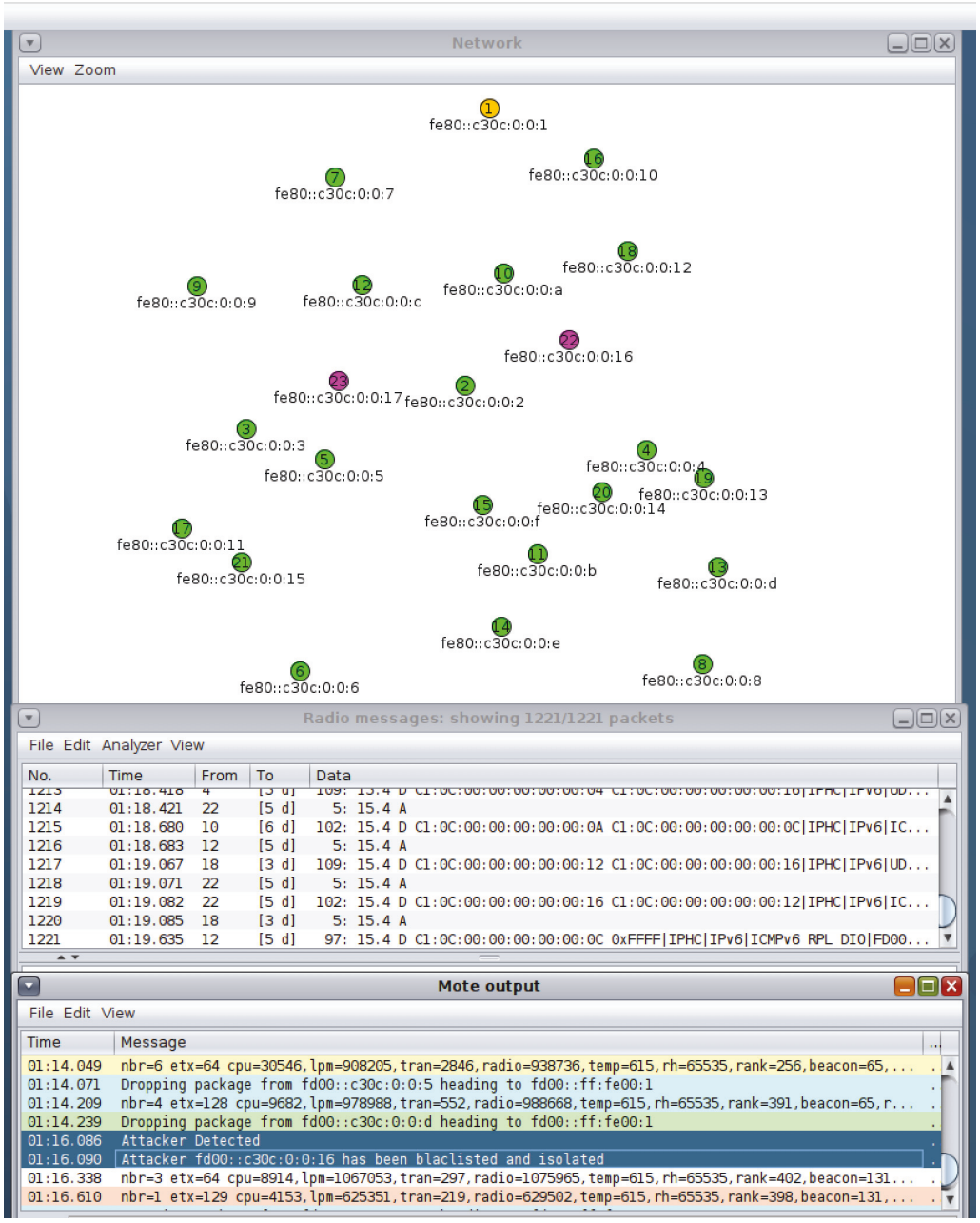


Figure 8.7 : IDS in operation

utes after attack initiation, the accuracy result dropped approximately 6 % for all of the attacks, as shown in Table 8.2. Although, the detection rate is relatively exceptionally high in this scenario, we can improve it by maintaining a blacklist of all of the malicious nodes. This ensures the same attacker node is isolated immediately. Figure 8.6 shows an example of attack implementation and the placement of the attacker's node shown in purple (nodes 17, 16), and Figure 8.7 shows the operation

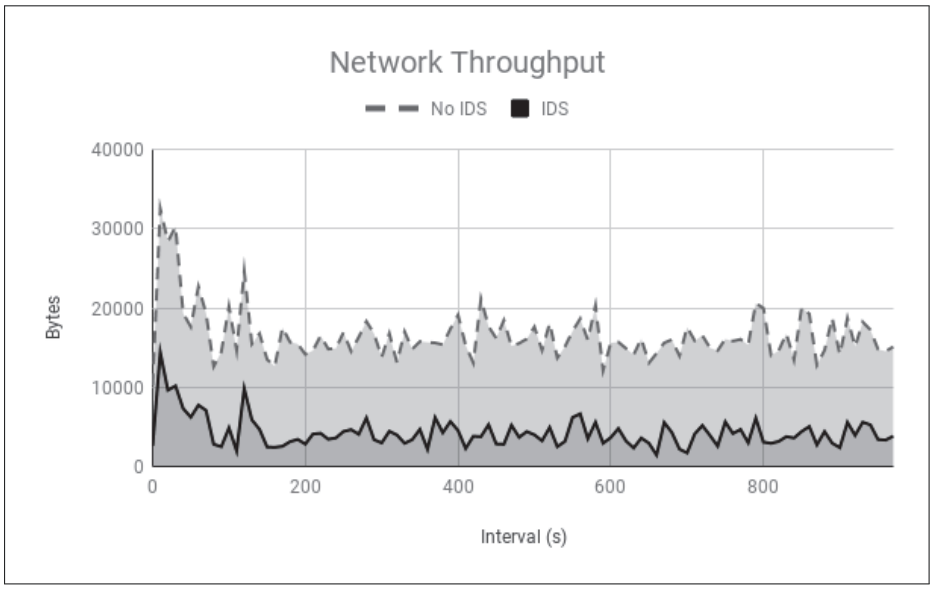


Figure 8.8 : Network throughput

of the IDS and how it is able to isolate the attacker nodes from the network. For full Cooja implementation, we have uploaded the entire Cooja IDS into GitHub in this link ("https://github.com/yss8166/sixlowpan_parser/") [123].

8.4.1 Network Performance Evaluation:

Table 8.3 : Simulation parameters

Parameter	Value
Number of nodes	21 nodes
Cluster head	1
Network area	100m x 100m
Size of packet	500
Transmission area	150m
Routing protocol	RPL
MAC protocol	CSMA
Simulation time	24 hrs

In this section, we measure network performance using two scenarios with and without the IDS. The network configuration should be the same in both scenar-

ios, which is summarised in Table 8.3. We measure two parameters in relation to network performance, network throughput and packet loss for the overall network for both scenarios with and without the IDS. As can be seen from Figure 8.8, the throughput decreased in the case where the IDS was introduced for the simulation of the blackhole and the selective forwarding attacks. This is because the IDS reduces the number of drop packets when the malicious node is isolated from the network. Although there are no substantial throughput changes for the nodes that are not affected by the attack, there is a significant improvement for the victim nodes. The packet loss for all attacks both with and without the IDS and without is summarized in Figure 8.9 and as can be seen, the effect of the IDS is significant across all of the attacks. The number of packet losses for the victim nodes decrease when the IDS is introduced. Such an improvement can be attributed to the blacklisting mechanism that we implemented to isolate the attacker node that disturbs the network health. The next section explores the overhead of the IDS in terms of energy consumption for a single node.

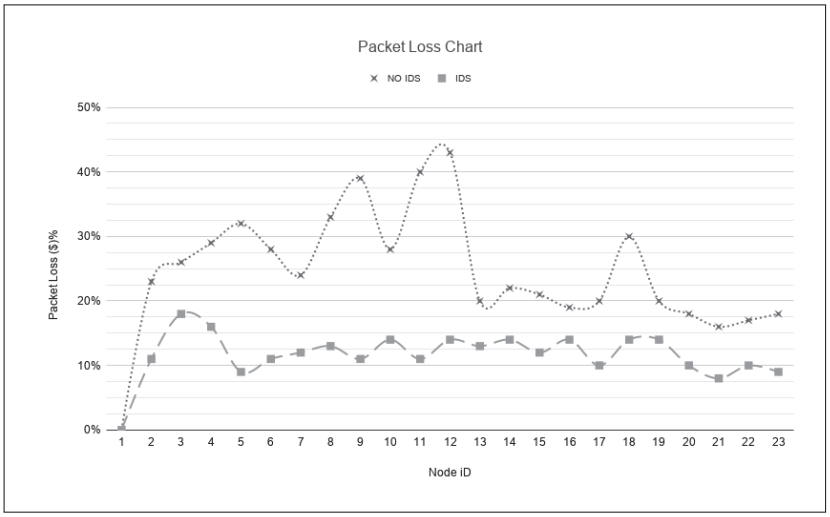


Figure 8.9 : packet loss chart

This is can be extended to include other statistics about the nodes in the network, and the rank of each node over time is summarized in 8.10. This shows how the number of ranks over time decreases when the IDS is activated.

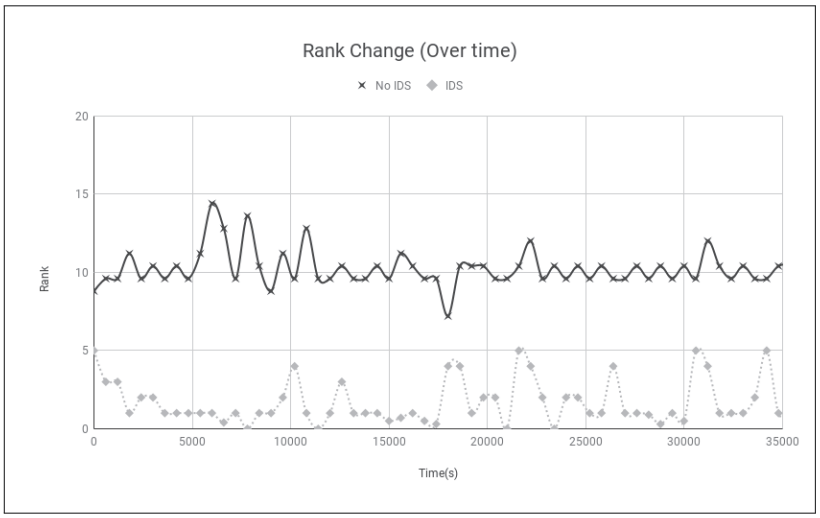
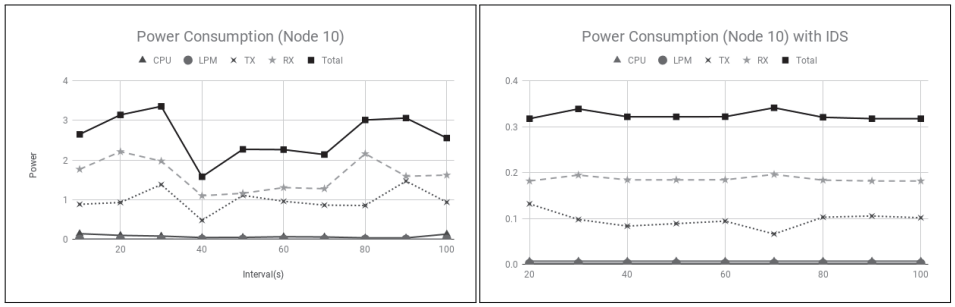


Figure 8.10 : Rank change over time (blackhole)

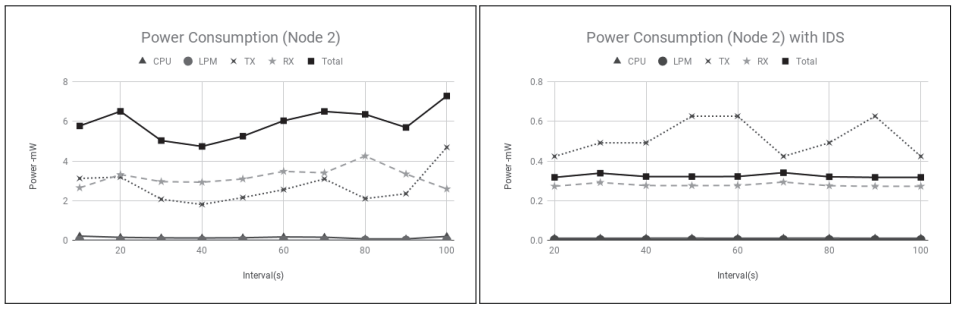
8.4.2 Signal Node Evaluation:

In this section of the evaluation, we monitor the performance of nodes 10 and 2 by looking at the battery indicators and evaluate their behavior when the IDS is activated. We evaluate all of the nodes in the network, and fortunately in Contiki OS, there are built-in modules called PowerTrace which allow us to monitor the power usage of each node. We implement this module in each node and monitor the battery performance in relation to the attacks with and without the system. However, we focus on node 10 and node 2 since they are the most affected by the attackers due to their close proximity to the attack.



(a) Node 10 power without IDS (b) Node 10 power without IDS

Figure 8.11 : Power consumption of node 10



(a) Node 2 power without IDS (b) Node 2 power with IDS

Figure 8.12 : Power consumption of node 2

Figures 8.12 and Figure 8.11 show the difference in power consumption for two cases with and without the IDS. As can be seen, the power consumption drops for both nodes when the IDS is introduced. The isolation and blacklisting of the attackers allowed the affected nodes to resume their RPL routing and network operations to normal behavior, thus reducing the power consumption and optimizing the duty cycle of each node in the network.

8.5 Web Interface to Monitor IDS-Performance

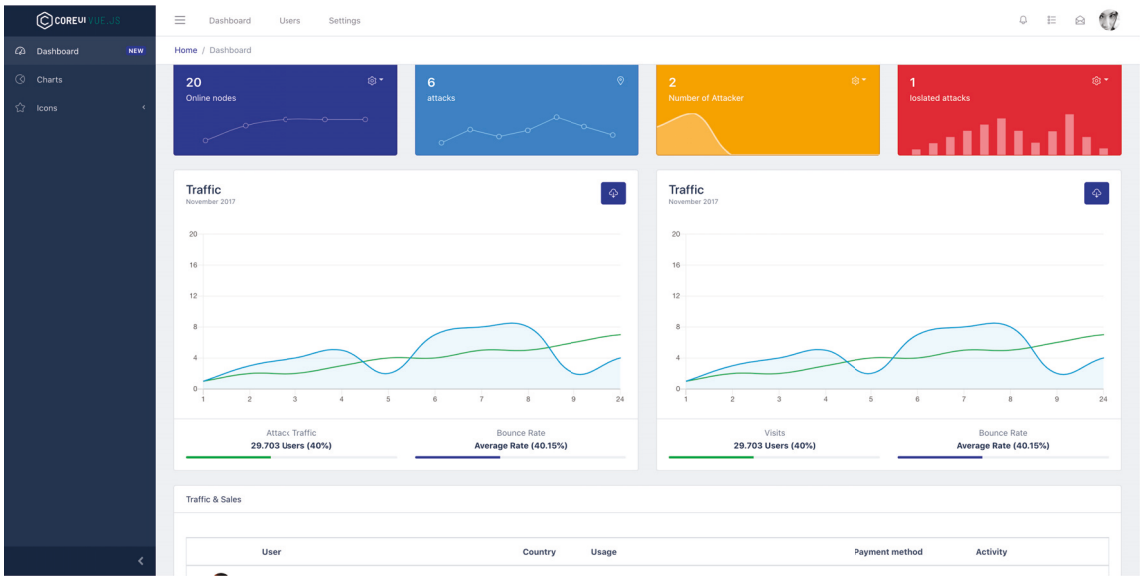


Figure 8.13 : Web monitor

The web application uses a set of values generated by the IDS Agent to display

a set of information in real-time regarding the status of the nodes in the network. Once an attack is detected, the web interface shows the exact attacker and how the system is able to isolate the node. The system also shows the current IDS node performance overhead. Moreover, it also handles all of the network administrator notifications. Figure 8.13 shows how the web interface is implemented. We use the Nodejs and MySQL database as the server-side to ensure real-time data streaming. To link between the database and the IDS, we utilize the REST-API functions to store the logs of all traffic analyzed whether it is attack data or regular traffic. When the attacker is detected, an alert is created containing the IP address and the type of attack and the victim node. This information is stored in the local storage (MySQL database) and is later retrieved by the REST-API to the web application front-end. This will allow us to compare the traffic volume ratio between the attack and no-attack traffic. The web front-end interface was built using HTML, CSS, bootstrap, Vuejs, and Nodejs and MySQL as back-end servers. This simple interface is just a demonstration of how we can integrate different third-party applications into the IDS.

8.6 Conclusion

In this chapter, we present an intelligent IDS solution implementation in a Contiki OS-based network. Different component implementations of the IoT-DDoS are discussed in this chapter. Moreover, building on Chapter 6 of the thesis, we implemented a real-time data collection and aggregation algorithm in the IoT network. We also present the feature extraction unit integration with the rest of the IDS. A comprehensive analysis of the implemented system is also presented, showing promising results for our IDS and its integration with the IoT network. Furthermore, in this chapter, we have shown how our IDS is able to detect the three attacks in the IoT environment which improves network performance and isolates the attacker nodes. Additionally, the second IDS is able to reduce node power consumption after the attacker node is isolated, proving that the IDS is able to detect and isolate attacker nodes.

Chapter 9

Recapitulation and Future Research Directions

9.1 Introduction

This chapter concludes this thesis by recalling its contribution and outlining suggested future work. The IoT is already playing a significant part in our daily life [61]. From smart home appliances like smart fridges and coffee machines to more complex applications like the ones we see in smart factories, all these applications have made our lives more efficient and easier. However, such rapid advancement in the IoT and its surrounding technologies presents risks and challenges. One of these challenges is security-related issues such as ensuring devices with sensitive information are secure and ensuring reliable security measures are implemented in these networks in consideration of their limited resource nature. An IoT-based DDoS is one of these security issues that are trending, which not only affects IoT networks, it also affects other sensitive networks. Solving DDoS attack issues in IoT networks is crucial to ensure a sustainable and reliable system. Therefore, a non-conventional security framework that takes into consideration the limitations associated with limited resource devices is a necessity. Furthermore, the advancement of AI and its applications opens new security opportunities for IoT industry research, and it is evident from the comprehensive review presented in Chapter 2 that the current research lacks a dedicated machine learning architecture that addresses security issues in IoT. Nonetheless, the current literature is rich with machine learning security applications across other kinds of networks like ad-hocs and WSN networks; however, as we presented in the shortcomings section in Chapter 2 and Chapter 3, the literature lacks a dedicated machine learning architecture. In this thesis, we present an intelligent machine learning platform for detecting DDoS attacks in IoT networks. As previously mentioned, this chapter concludes this thesis and is organized as fol-

lows: in section 9.2, the issues discussed in this thesis are presented. Based on these issues, the objective and contributions of this thesis are listed in section 9.3. Finally this chapter is concluded in section 9.4

9.2 Problems Addressed in This Thesis

This thesis addresses the problem of DDoS attacks in IoT networks using intelligent machine learning methods. In Chapter 2, the literature was thoroughly investigated and the research issues addressed in this thesis were summarized as follows:

1. Propose and develop a real-time dataset generation framework for IoT networks to ensure up-to-date IoT datasets tailored towards IoT security research.

2. Propose machine learning algorithms that are capable of detecting malicious attacks in IoT networks. In this thesis, we evaluated and compared the performance of the SVM method, neural networks, and isolation tree algorithms. Furthermore, the performance of each method against blackhole, selective forwarding, and flooding attacks was analyzed.

3. Propose and develop an IoT IDS (IoT-DDoS) for detecting three malicious attacks (blackhole, selective forwarding, and flooding attacks) using a support vector machine (SVM) algorithm.

9.3 Contribution of This Thesis to the Existing Literature

Based on the identified research issues and gaps, this thesis develops an intelligent machine learning framework for detecting distributed details of service attacks in IoT networks. The four main contributions of this thesis are summarized as follows:

9.3.1 Contribution 1: Comprehensive State-of-the-art Survey of the Existing Literature

This thesis comprehensively reviews the existing literature in the area of DDoS attack detection in IoT, machine learning attack detection in IoT, and SVM use for

detecting selective forwarding, blackhole, and flooding attacks. To the best of our knowledge, there is no comprehensive review of the literature in this area. After a comprehensively investigation of the literature, it was divided into four categories based on their solutions, methodologies and target domains:

1. Protocol-based approaches for attack detection in IoT.
2. IDS-based methods used for attack analysis and detection in IoT.
3. Trust-based approaches used for securing communication between entities in IoT.
4. Machine learning approaches used for attack detection in general.

Each of these approaches was investigated thoroughly, and the shortcomings of each approach were identified. Furthermore, based on the extensive analysis of the literature, the research gaps and research questions were formed and identified, and we submitted the results to a Q1 Journal and the manuscript is in the second round review stage [2]

9.3.2 Contribution 2: A Framework for Attack Detection in IoT Using Intelligent Machine learning Methods

This thesis examines how IoT networks work differently to other traditional networks due to their heterogeneous nature and the complexity of their implementation. Therefore, building any security solution for IoT networks needs to take into consideration all of the factors such as limited resources and the heterogeneity of the network and devices. To address these limitations in the literature, a comprehensive machine learning framework was proposed to detect DDoS attack in RPL/6LoWPAN based IoT networks intelligently. The main features of the proposed framework are summarized as follows:

1. Real-time data collection: this feature is used to intelligently collect the data based on the network configuration. This allows more dynamic data acquisition, ensuring up-to-date datasets tailored for IoT usage.

2. Attack detection unit: This is the heart of the framework where all the attack analysis and detection take place. This unit is implemented at an edge IoT node, which is the 6BR router since it has more power compared to IoT constrained devices.

3. IDS agent: This is the root node in the network, sometimes called the sink node, which is responsible for secure communication with the attack detection unit in bi-directional communications. Furthermore, the IDS agent is responsible for ensuring the RPL network topology is functioning as expected. Also, the IDS agent is responsible for alarm generation in case there are any attacks.

4. Third-party API: This is responsible for storing the results of the attack detection units and stores it in a local repository which can be accessed later by third-party applications if needed.

A detailed explanation of the framework with its related implementation algorithms is described in Chapter 5 of this thesis. To the best of our knowledge, this is the first time such a detailed framework has been presented to solve the issue of DDoS attack detection in IoT using machine learning. Furthermore, the result of this contribution has been published in one of the top journals in the field, Future Generation Computer Systems (Q1 journal) [124].

9.3.3 Contribution 3: A real-time Framework for Dataset Generation in IoT Environments

To build any machine learning model, the need for up-to-date and accurate data is crucial. Therefore, due to the lack of IoT-related datasets tailored to detect IoT-specific attacks, we propose a new framework for data generation and collection for IoT environments. This framework allows for more precise and accurate protocol representation for attack detection and testing scenarios. Chapter 6 discusses in detail the intelligent real-time data collection framework, and how the data generation and gathering are handled. To ensure good quality data, the following algorithms were proposed:

- Attack Algorithms: Blackhole, selective forwarding, and flooding attack algorithms are proposed which represent a real attack implementation in real-world scenarios. These algorithms are crucial to ensure accurate attack representation in the final dataset.

- Capturing algorithm: This algorithm is responsible for capturing the traffic from sniffer nodes without compromising the scalability of the network. Furthermore, it ensures complete network coverage by analyzing the RSSI node signals.

- Queueing algorithm: The queueing algorithm is responsible for accurately aggregating data without any duplication when we have multiple data inputs. The algorithms were designed, taking into consideration the dynamically growing IoT network.

To the best of our knowledge, this the first time a real-time dataset generation and collection framework has been presented to tackle DDoS attacks in IoT networks.

9.3.4 Contribution 4: Intelligent Machine Learning Methodology for DDoS Attack Detection in IoT

The fourth contribution comprises three machine learning algorithms used in combination with the DDoS-IoT datasets generated in Chapter 6. Chapter 7 details all of the mathematical representations of each algorithm, with rigorous and extensive training and testing of SVMs, neural networks and the isolation tree machine learning models. The aforementioned machine learning methods were implemented and evaluated comprehensively to detect three kinds of DDoS attacks, blackhole, selective forwarding, and the flooding attacks. Furthermore, a detailed process was developed to ensure consistent results across all of the machine learning models and to avoid any biased implementations. To the best of our knowledge, none of the existing research explores the use of machine learning models with intelligently gathered datasets to detect DDoS attacks in IoT.

9.3.5 Contribution 5: Implementation and Evaluation of the Proposed IDS Framework in a Semi-Real IoT Environment

To evaluate the proposed attack detection IDS framework presented in Chapters 5, 6, and 7, we followed specific matrices and benchmarks to ensure accurate result representations for each scenario presented in Chapter 8. Furthermore, the detailed implementation of the IDS, including testing and evaluation under different scenarios, was presented in Chapter 8. To build the entire system, we used Cooja [77] as a simulation platform using Contiki OS as an IoT operating system, which was the base for our development. For the real-time data collection unit and parser, we used Python 3.7 with a TShark framework. Also, the base of the IDS detection unit was developed using Python and the sklearn library. The detailed implementation of the various components of the framework was presented in Chapter 8. The detailed process of building the entire framework was presented sequentially from Chapter 5 to Chapter 8. We present some key findings of the whole system as follows:

1. We found our proposed approach for real-time data collection was able to generate up-to-date datasets with different sets of IoT attack patterns, ensuring the correct and precise representation of IoT traffic to be used for machine learning purposes.

2. We found after rigorous testing that the machine learning model which best fits the collected dataset is the SVM model followed by the random forest.

3. We found that the introduction of our IDS did not greatly affect the network throughput and performance. As a matter of fact, the system was able to reduce network throughput when the network was under attack.

4. We found the neural network methods with a single hidden layer performed poorly when combined with the DDoS-IoT dataset.

5. We found the selected SVM method used in our IDS was able to correctly and effectively detect the three attacks (blackhole, selective forwarding, and the flooding attack).

9.4 Conclusion and Future Work

IoT is growing at a fast pace, involving sectors and domains that had previously never been considered. This includes but is not limited to factories, agriculture, cities, and transportation. This wide exploitation of such technology and the rapid adaptions in a wide variety of sectors bring many security challenges and issues. Among them is the DDoS attack that affects the availability of resources, causing financial and resource loss. This thesis presents an intelligent framework for DDoS attack detection using machine learning, which addresses the issue of the lack of a dataset for machine learning evolution and explores three DDoS attacks in IoT, namely selective forwarding, blackhole and flooding attack. The work done in this thesis has been published in peer-reviewed conferences and journals. At the time of writing this thesis, one paper has been published in the FGCS Q1 [124] journal, and another paper is undergoing a second-round review for a WWW Q1 journal. Furthermore, two conference papers have also been published [97] [127]. The list of publications as a result of this research is presented at the beginning of this thesis. Although extensive research has been undertaken as part of this thesis, there is still room for improvement in strengthening the system proposed in this thesis. To further enhance the outcome of this thesis, we propose the following future research directions:

- To further explore other machine learning models and expand the number of attacks that can be detected by the system. The current version of our IDS analyzes and detects three kinds of IoT DDoS attacks, however, in the future , we aim to expand this to include more attacks which can be detected by our IDS. We designed our IDS to be future proof by adding the third-party REST-API and we can expand the IDS functionality to add more attacks and tools as needed.

- To develop an additional mechanism that allows reinforcement learning into the machine learning model, allowing it to adapt to more new unknown attacks by deeply analyzing the IoT traffic. At the current state of our IDS is based

on a machine learning model, the framework is designed in such a way that it allows us to change it to include reinforcement in the future, enabling it to be more sophisticated when the system frequently detects unknown attacks. Such future thinking allows us to improve the way the IDS functions and more features means it will eventually handle a wider range of unknown attacks.

- To develop intelligent distributed IDS agents across all of the nodes, ensuring there is no need for external sniffer nodes. The current version of the IDS is semi-centralized where all of the nodes undertake some kind of IDS processing based on the IDS agent, but this depends mainly on the IDS core function installed on the 6BR router due to the limitations associated with IoT devices. However, in the future, we aim to make the IDS distributed so it does not depend on the centralized 6BR router. Although such an idea is challenging due to the resource constraints of IoT devices,with the development of the concept of FOG computing, this challenge can be overcome in the future when fog computing is mature enough to handle such heavy processing.

Bibliography

[1] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, 2017.

[2] Yahya Al-Hadhrami and Farookh Khadeer Hussain. Ddos attacks in iot networks: A comprehensive systematic literature review. *World Wide Web*, 2020.

[3] Minhaj Ahmad Khan and Khaled Salah. Iot security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, 82:395–411, 2018.

[4] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[5] Faraz Idris Khan, Taeshik Shon, Taekkyeun Lee, and Kihyung Kim. Wormhole attack prevention mechanism for rpl based lln network. In *2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 149–154. IEEE, 2013.

[6] Joakim Eriksson, Fredrik Österlind, Niclas Finne, Adam Dunkels, Nicolas Tsiftes, and Thiemo Voigt. Accurate network-scale power profiling for sensor network simulators. In *European Conference on Wireless Sensor Networks*, pages 312–326. Springer, 2009.

[7] Sandeep Dhawan. Information and data security concepts, integrations,

limitations and future. *International Journal of Advanced Information Science and Technology (IJAIST)*, 3(9):9, 2014.

[8] Mian Ahmad Jan and Muhammad Khan. Denial of service attacks and their countermeasures in wsn. *IRACST–International Journal of Computer Networks and Wireless Communications (IJCNWC)*, 3, 2013.

[9] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, and Hui-Ying Du. Research on the architecture of internet of things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, volume 5, pages V5–484. IEEE, 2010.

[10] Tim Winter. Rpl: Ipv6 routing protocol for low-power and lossy networks. 2012.

[11] Pavan Pongle and Gurunath Chavan. A survey: Attacks on rpl and 6lowpan in iot. In *2015 International conference on pervasive computing (ICPC)*, pages 1–6. IEEE, 2015.

[12] Pavan Pongle and Gurunath Chavan. Real time intrusion and wormhole attack detection in internet of things. *International Journal of Computer Applications*, 121(9), 2015.

[13] Prabhakaran Kasinathan, Claudio Pastrone, Maurizio A Spirito, and Mark Vinkovits. Denial-of-service detection in 6lowpan based internet of things. In *2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)*, pages 600–607. IEEE, 2013.

[14] Jin Ye, Xiangyang Cheng, Jian Zhu, Luting Feng, and Ling Song. A ddos attack detection method based on svm in software defined network. *Security and Communication Networks*, 2018, 2018.

[15] Anuj Sehgal, Anthéa Mayzaud, Rémi Badonnel, Isabelle Chrisment, and Jürgen Schönwälder. Addressing dodag inconsistency attacks in rpl networks. In *2014 Global Information Infrastructure and Networking Symposium (GIIS)*, pages 1–8. IEEE, 2014.

[16] Avijit Mathur, Thomas Newe, and Muzaffar Rao. Defence against black hole and selective forwarding attacks for medical wsns in the iot. *Sensors*, 16(1):118, 2016.

[17] Karan Verma, Halabi Hasbullah, and Ashok Kumar. An efficient defense method against udp spoofed flooding traffic of denial of service (dos) attacks in vanet. In *2013 3rd IEEE International Advance Computing Conference (IACC)*, pages 550–555. IEEE, 2013.

[18] David Airehrour, Jairo Gutierrez, and Sayan Kumar Ray. A lightweight trust design for iot routing. In *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 552–557. IEEE, 2016.

[19] Wan Haslina Hassan et al. Current research on internet of things (iot) security: A survey. *Computer Networks*, 148:283–294, 2019.

[20] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, and Hicham Lakhlef. Internet of things security: A top-down survey. *Computer Networks*, 141:199–221, 2018.

[21] Aakanksha Tewari and BB Gupta. Security, privacy and trust of different layers in internet-of-things (iots) framework. *Future Generation Computer Systems*, 2018.

[22] Yang Lu and Li Da Xu. Internet of things (iot) cybersecurity research: a review of current research topics. *IEEE Internet of Things Journal*, 2018.

[23] Arbia Riahi Sfar, Enrico Natalizio, Yacine Challal, and Zied Chtourou. A roadmap for security challenges in the internet of things. *Digital Communications and Networks*, 4(2):118–137, 2018.

[24] Arsalan Mosenia and Niraj K Jha. A comprehensive study of security of internet-of-things. *IEEE Transactions on Emerging Topics in Computing*, 5(4):586–602, 2017.

[25] Kewei Sha, Wei Wei, T Andrew Yang, Zhiwei Wang, and Weisong Shi. On security challenges and open issues in internet of things. *Future Generation Computer Systems*, 83:326–337, 2018.

[26] Nadia Chaabouni, Mohamed Mosbah, Akka Zemmari, Cyrille Sauvignac, and Parvez Faruki. Network intrusion detection for iot security based on learning techniques. *IEEE Communications Surveys & Tutorials*, 2019.

[27] Fadele Ayotunde Alaba, Mazliza Othman, Ibrahim Abaker Targio Hashem, and Faiz Alotaibi. Internet of things security: A survey. *Journal of Network and Computer Applications*, 88:10–28, 2017.

[28] Lirong Zheng, Hui Zhang, Weili Han, Xiaolin Zhou, Jing He, Zhi Zhang, Yun Gu, Junyu Wang, et al. Technologies, applications, and governance in the internet of things. *Internet of things-Global technological and societal trends. From smart environments and spaces to green ICT*, 2011.

[29] Tuhin Borgohain, Uday Kumar, and Sugata Sanyal. Survey of security and privacy issues of internet of things. *arXiv preprint arXiv:1501.02211*, 2015.

[30] Tapalina Bhattasali, Rituparna Chaki, and Sugata Sanyal. Sleep deprivation attack detection in wireless sensor network. *arXiv preprint arXiv:1203.0231*, 2012.

[31] Y. Wang, G. Attebury, and B. Ramamurthy. A survey of security issues in wireless sensor networks. *IEEE Communications Surveys Tutorials*, 8(2):2–23, Second 2006.

[32] René Hummen, Jens Hiller, Hanno Wirtz, Martin Henze, Hossein Shafagh, and Klaus Wehrle. 6lowpan fragmentation attacks and mitigation mechanisms. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pages 55–66. ACM, 2013.

[33] Parmar Amish and VB Vaghela. Detection and prevention of wormhole attack in wireless sensor network using aomdv protocol. *Procedia computer science*, 79:700–707, 2016.

[34] Minzhao Lyu, Dainel Sherratt, Arunan Sivanathan, Hassan Habibi Gharakheili, Adam Radford, and Vijay Sivaraman. Quantifying the reflective ddos attack capability of household iot devices. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 46–51. ACM, 2017.

[35] Ghada Glissa and Aref Meddeb. 6lowpsec: An end-to-end security protocol for 6lowpan. *Ad Hoc Networks*, 82:100–112, 2019.

[36] Linus Wallgren, Shahid Raza, and Thiemo Voigt. Routing attacks and countermeasures in the rpl-based internet of things. *International Journal of Distributed Sensor Networks*, 9(8):794326, 2013.

[37] M Shamim Hossain, Ghulam Muhammad, Sk Md Mizanur Rahman, Wadood Abdul, Abdulhameed Alelaiwi, and Atif Alamri. Toward end-to-end biomet rics-based security for iot infrastructure. *IEEE Wireless Communications*, 23(5):44–51, 2016.

[38] Ghada Glissa and Aref Meddeb. 6lowpan multi-layered security protocol based on ieee 802.15. 4 security features. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 264–269. IEEE, 2017.

[39] Cong Pu and Sunho Lim. A light-weight countermeasure to forwarding misbehavior in wireless sensor networks: design, analysis, and evaluation. *IEEE Systems Journal*, 12(1):834–842, 2016.

[40] Karel Heurtefeux, Ochirkhand Erdene-Ochir, Nasreen Mohsin, and Hamid Menouar. Enhancing rpl resilience against routing layer insider attacks. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pages 802–807. IEEE, 2015.

[41] Mahmud Hossain, Yasser Karim, and Ragib Hasan. Secupan: A security scheme to mitigate fragmentation-based network attacks in 6lowpan. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pages 307–318. ACM, 2018.

[42] Amit Dvir, Levente Buttyan, et al. Vera-version number and rank authentication in rpl. In *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*, pages 709–714. IEEE, 2011.

[43] Heiner Perrey, Martin Landsmann, Osman Ugus, Thomas C Schmidt, and Matthias Wählisch. Trail: Topology authentication in rpl. *arXiv preprint arXiv:1312.0984*, 2013.

[44] Fatma Gara, Leila Ben Saad, and Rahma Ben Ayed. An intrusion detection system for selective forwarding attack in ipv6-based mobile wsns. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 276–281. IEEE, 2017.

[45] Robert L Pihl. The sequential probability ratio test. *History 9*, page 1, 1998.

[46] Christian Cervantes, Diego Poplade, Michele Nogueira, and Aldri Santos. Detection of sinkhole attacks for supporting secure routing on 6lowpan for internet of things. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 606–611. IEEE, 2015.

[47] Zeeshan Ali Khan and Peter Herrmann. A trust based distributed intrusion detection mechanism for internet of things. In *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pages 1169–1176. IEEE, 2017.

[48] Faiza Medjek, Djamel Tandjaoui, Imed Romdhani, and Nabil Djedjig. A trust-based intrusion detection system for mobile rpl based networks. In *2017 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 735–742. IEEE, 2017.

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

[50] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4):2347–2376, 2015.

[51] Audun Jøsang. A logic for uncertain probabilities. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 9(3):279–311, June 2001.

[52] Ray Chen, Fenye Bao, and Jia Guo. Trust-based service management for social internet of things systems. *IEEE transactions on dependable and secure computing*, 13(6):684–696, 2015.

[53] M Surendar and A Umamakeswari. Indres: An intrusion detection and response system for internet of things with 6lowpan. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 1903–1908. IEEE, 2016.

[54] Anhtuan Le, Jonathan Loo, Kok Chai, and Mahdi Aiash. A specification-based ids for detecting attacks on rpl-based network topology. *Information*, 7(2):25, 2016.

[55] Shahid Raza, Linus Wallgren, and Thiemo Voigt. Svelte: Real-time intrusion detection in the internet of things. *Ad hoc networks*, 11(8):2661–2674, 2013.

[56] Vinh Hoa La, Raul Fuentes, and Ana R Cavalli. A novel monitoring solution for 6lowpan-based wireless sensor networks. In *2016 22nd Asia-Pacific Conference on Communications (APCC)*, pages 230–237. IEEE, 2016.

[57] Qussai Yaseen, Firas AlBalas, Yaser Jararweh, and Mahmoud Al-Ayyoub. A fog computing based system for selective forwarding detection in mobile wireless sensor networks. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, pages 256–262. IEEE, 2016.

[58] Tariqahmad Sherasiya and Hardik Upadhyay. Intrusion detection system for internet of things. *Int. J. Adv. Res. Innov. Ideas Educ.(IJARIIE)*, 2(3), 2016.

[59] Krushang Sonar and Hardik Upadhyay. An approach to secure internet of things against ddos. In *Proceedings of International Conference on ICT for Sustainable Development*, pages 367–376. Springer, 2016.

[60] Daniele Midi, Antonino Rullo, Anand Mudgerikar, and Elisa Bertino. Kalis—a system for knowledge-driven adaptable intrusion detection for the internet of things. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 656–666. IEEE, 2017.

[61] Philokypros Ioulianou, Vasileios Vasilakis, Ioannis Moscholios, and Michael Logothetis. A signature-based intrusion detection system for the internet of things. *Information and Communication Technology Form*, 2018.

[62] R Stephen and L Arockiam. Intrusion detection system to detect sinkhole attack on rpl protocol in internet of things. *International Journal of Electrical Electronics and Computer Science*, 4(4):16–20, 2017.

[63] Hamid Bostani and Mansour Sheikhan. Hybrid of anomaly-based and specification-based ids for internet of things using unsupervised opf based on mapreduce approach. *Computer Communications*, 98:52–71, 2017.

[64] In Lee and Kyoochun Lee. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4):431–440, 2015.

[65] Qussai Yaseen, Firas Albalas, Yaser Jararwah, and Mahmoud Al-Ayyoub. Leveraging fog computing and software defined systems for selective

forwarding attacks detection in mobile wireless sensor networks. *Transactions on Emerging Telecommunications Technologies*, 29(4):e3183, 2018.

[66] Rajesh Shrivastava, Chittaranjan Hota, and Prashast Shrivastava. Protection against code exploitation using rop and check-summing in iot environment. In *2017 5th International Conference on Information and Communication Technology (ICoIC7)*, pages 1–6. IEEE, 2017.

[67] Chittaranjan Hota, Rajesh Kumar Shrivastava, and Shivangi Shipra. Tamper-resistant code using optimal rop gadgets for iot devices. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 570–575. IEEE, 2017.

[68] Nima Namvar, Walid Saad, Niloofar Bahadori, and Brian Kelley. Jamming in the internet of things: A game-theoretic perspective. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

[69] Xiao Tang, Pinyi Ren, and Zhu Han. Jamming mitigation via hierarchical security game for iot communications. *IEEE Access*, 6:5766–5779, 2018.

[70] Yoonyoung Sung, Sookyoung Lee, and Meejeong Lee. A multi-hop clustering mechanism for scalable iot networks. *Sensors*, 18(4):961, 2018.

[71] Rwan Mahmoud, Tasneem Yousuf, Fadi Aloul, and Imran Zualkernan. Internet of things (iot) security: Current status, challenges and prospective measures. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 336–341. IEEE, 2015.

[72] SJ Stolfo et al. Kdd cup 1999 dataset. *UCI KDD repository. http://kdd. ics. uci. edu*, 1999.

[73] Gideon Creech and Jiankun Hu. Generation of a new ids test dataset: Time to retire the kdd collection. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 4487–4492. IEEE, 2013.

[74] Carson Brown, Alex Cowperthwaite, Abdulrahman Hijazi, and Anil Somayaji. Analysis of the 1999 darpa/lincoln laboratory ids evaluation data with netadhict. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–7. IEEE, 2009.

[75] Philippe Owezarski. A database of anomalous traffic for assessing profile based ids. In *International Workshop on Traffic Monitoring and Analysis*, pages 59–72. Springer, 2010.

[76] Gregory Kipper. Chapter 2-the types of augmented reality. *Augmented Reality*, pages 29–50, 2013.

[77] Fredrik Österlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. In *First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, 2006.

[78] Latha Tamilselvan and V Sankaranarayanan. Prevention of blackhole attack in manet. In *The 2nd international conference on wireless broadband and ultra wideband communications (AusWireless 2007)*, pages 21–21. IEEE, 2007.

[79] Geoff Mulligan. The 6lowpan architecture. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 78–82. ACM, 2007.

[80] Katherine Heller, Krysta Svore, Angelos D Keromytis, and Salvatore Stolfo. One class support vector machines for detecting anomalous windows registry accesses. 2003.

[81] Qiankun Song and Jinde Cao. Stability analysis of cohen–grossberg neural network with both time-varying and continuously distributed delays. *Journal of Computational and Applied Mathematics*, 197(1):188–203, 2006.

[82] Mary Jo Aspinall. Use of a decision tree to improve accuracy of diagnosis. *Nursing Research*, 28(3):182–185, 1979.

[83] Nicola Accettura, Luigi Alfredo Grieco, Gennaro Boggia, and Pietro Camarda. Performance analysis of the rpl routing protocol. In *2011 IEEE International Conference on Mechatronics*, pages 767–772. IEEE, 2011.

[84] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.

[85] Frada Burstein and Shirley Gregor. The systems development or engineering approach to research in information systems: An action research perspective. In *Proceedings of the 10th Australasian Conference on Information Systems*, pages 122–134. Victoria University of Wellington, New Zealand, 1999.

[86] Todd D Jick. Mixing qualitative and quantitative methods: Triangulation in action. *Administrative science quarterly*, 24(4):602–611, 1979.

[87] Dong Seong Kim, Ha-Nam Nguyen, and Jong Sou Park. Genetic algorithm to improve svm based network intrusion detection system. In *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*, volume 2, pages 155–158. IEEE, 2005.

[88] Hossein Gharaee and Hamid Hosseinvand. A new feature selection ids based on genetic algorithm and svm. In *2016 8th International Symposium on Telecommunications (IST)*, pages 139–144. IEEE, 2016.

[89] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[90] Mark A Friedl and Carla E Brodley. Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment*, 61(3):399–409, 1997.

[91] George Oikonomou. SenSniff: Live Traffic Capture and Sniffer for IEEE 802.15.4 networks., 2017.

[92] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.

[93] Wenke Lee and Salvatore Stolfo. Data mining approaches for intrusion detection. 1998.

[94] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6. IEEE, 2009.

[95] S Revathi and A Malathi. A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection. *International Journal of Engineering Research & Technology (IJERT)*, 2(12):1848–1853, 2013.

[96] Mohammad Khubeb Siddiqui and Shams Naahid. Analysis of kdd cup 99 dataset using clustering based data mining. *International Journal of Database Theory and Application*, 6(5):23–34, 2013.

[97] Yahya Al-Hadhrami, Nasser Al-Hadhrami, and Farookh Khadeer Hussain. Data exportation framework for iot simulation based devices. In *International Conference on Network-Based Information Systems*, pages 212–222. Springer, 2019.

[98] Joakim Eriksson, Adam Dunkels, Niclas Finne, Fredrik Osterlind, and Thiemo Voigt. Mspsim–an extensible simulator for msp430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, volume 118, 2007.

[99] George Reese. *Database Programming with JDBC and JAVA*. " O'Reilly Media, Inc.", 2000.

[100] Paul DuBois and Michael Foreword By-Widenius. *MySQL*. New riders publishing, 1999.

[101] Marie-Paule Uwase, Nguyen Thanh Long, Jacques Tiberghien, Kris Steenhaut, and Jean-Michel Dricot. Outdoors range measurements with zolertia z1 motes and contiki. In *Real-world wireless sensor networks*, pages 79–83. Springer, 2014.

[102] IEEE. Ieee standard for low-rate wireless networks - amendment 5: Enabling/updating the use of regional sub-ghz bands. *IEEE Std 802.15.4v-2017 (Amendment to IEEE Std 802.15.4-2015, as amended by IEEE Std 802.15.4n-2016, IEEE Std 802.15.4q-2016, IEEE Std 802.15.4u-2016, and IEEE Std 802.15.4t-2017)*, pages 1–35, June 2017.

[103] Saidi Nabiha, Khaled Zaatouri, Walid Fajraoui, and Tahar Ezzeddine. New design of low power consumption mote in wireless sensor network. *Power*, 20(15):3, 2015.

[104] Adam Dunkels, Oliver Schmidt, Niclas Finne, Joakim Eriksson, Fredrik Österlind, and Nicolas Tsiftesand Mathilde Durvy. The contiki os: The operating system for the internet of things. *Online], at http://www. contikios. org*, 605, 2011.

[105] JL Romkey. Nonstandard for transmission of ip datagrams over serial lines: Slip. 1988.

[106] Chipcon CC2420. 2.4 ghz ieee 802.15. 4/zigbee-ready rf transceiver. *Chipcon Products from Texas Instruments*, 2006.

[107] Steven McCanne and Van Jacobson. The bsd packet filter: A new architecture for user-level packet capture. In *USENIX winter*, volume 46, 1993.

[108] Philippe Biondi et al. Scapy. `https://github.com/secdev/scapy`, 2011.

[109] Wei Zhou, Junhao Wen, Qingyu Xiong, Min Gao, and Jun Zeng. Svm-tia a shilling attack detection method based on svm and target item analysis in recommender systems. *Neurocomputing*, 210:197–205, 2016.

[110] Kun-Lun Li, Hou-Kuan Huang, Sheng-Feng Tian, and Wei Xu. Improving one-class svm for anomaly detection. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693)*, volume 5, pages 3077–3081. IEEE, 2003.

[111] Jungtaek Seo, Cheolho Lee, Taeshik Shon, Kyu-Hyung Cho, and Jongsub Moon. A new ddos detection model using multiple svms and tra. In *International Conference on Embedded and Ubiquitous Computing*, pages 976–985. Springer, 2005.

[112] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[113] R Tyrrell Rockafellar. Lagrange multipliers and optimality. *SIAM review*, 35(2):183–238, 1993.

[114] BD Craven. A modified wolfe dual for weak vector minimization. *Numerical Functional Analysis and Optimization*, 10(9-10):899–907, 1989.

[115] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[116] Jihyun Kim, Jaehyun Kim, Huong Le Thi Thu, and Howon Kim. Long short term memory recurrent neural network classifier for intrusion detection. In *2016 International Conference on Platform Technology and Service (PlatCon)*, pages 1–5. IEEE, 2016.

[117] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. Applying convolutional neural network for network intrusion detection. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1222–1228. IEEE, 2017.

[118] Sankar K Pal and Sushmita Mitra. Multilayer perceptron, fuzzy sets, classifiaction. 1992.

[119] Kuo-lin Hsu, Hoshin Vijai Gupta, and Soroosh Sorooshian. Artificial neural network modeling of the rainfall-runoff process. *Water resources research*, 31(10):2517–2530, 1995.

[120] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[121] Ting Zhi, Hongbin Luo, and Ying Liu. A gini impurity-based interest flooding attack defence mechanism in ndn. *IEEE Communications Letters*, 22(3):538–541, 2018.

[122] K Sklearn and Folds Cross-Validator. Available online: http://scikit-learn. org/stable/modules/generated/sklearn. model_selection. *KFold. html (accessed on 24 April 2018)*, 2018.

[123] Yahya Al-Hadhrami. Iot-ddos a seucrity based iot dataset. `https://github.com/yss8166/sixlowpan\_parser/`, 2019.

[124] Yahya Al-Hadhrami and Farookh Khadeer Hussain. Real time dataset generation framework for intrusion detection systems in iot. *Future Generation Computer Systems*, 108:414 – 423, 2020.

[125] Fredrik Österlind, Joakim Eriksson, and Adam Dunkels. Cooja timeline: a power visualizer for sensor network simulation. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 385–386. ACM, 2010.

[126] Miguel Grinberg. *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[127] Yahya Al-Hadhrami and Farookh Khadeer Hussain. A machine learning architecture towards detecting denial of service attack in iot. In *Conference on Complex, Intelligent, and Software Intensive Systems*, pages 417–429. Springer, 2019.