

## “Securing IoT Devices in Wireless Networks for Detection and Mitigation of MiTM and DDoS Attacks”

**Researcher:**

**Sara Al-Salouli**

## GRATITUDE AND DEDICATION

To our beloved and wonderful families, we dedicate this graduation project. You have been an incomparable support at every step of our academic journey. We have been fortunate to have individuals like you who encourage us and support us in all aspects, morally and financially. We are grateful to you for the trust you have placed in us and for your encouragement that has given us the strength to reach and overcome this academic stage, and we promise to continue striving for success in honor of your unfailing support, sincere love and gratitude to you.

## ACKNOWLEDGEMENT

Firstly, we would like to thank ALLAH for giving us resources and ability to write this project.

We would like to thank our supervisor Dr. for his invaluable guidance and assistance throughout the period of project development. Without his time and effort, this may not have been possible.

Finally, we would like to express special thanks to our lovely families for believing in us to further our education. Their endless trust, support and encouragement as well as advices helping us to continue this journey.

## ABSTRACT

This research aims to solve the security challenges faced by Internet of Things (IoT) devices in software-defined networking (SDN) environments, with a particular focus on mitigating man-in-the-middle and distributed denial-of-service (DDoS) attacks. The proposed approach for DDOS involves generating a dataset from a data stream in the network, which is then trained and tested using machine learning algorithms such as KNN, SVM, RF, DT, and NB. The goal is to identify the algorithm that has the highest accuracy resulting from Training and testing of the target data, which is found to be Random Forest (RF), which we will use for forecasting and data analysis purposes, and Random Forest (RF) will be used in the proposed model to detect and mitigate Distributed Denial of Service (DDoS) attacks. Also to enhance security To protect against ARP spoofing attacks, lookup provides ARP with a mechanism to detect this type of attack. The mechanism includes the following steps: extracting packet information, learning MAC-to-Port assignments, and checking if the packet is an ARP request. If it is an ARP request, it will be stored Set IP-MAC. If mitigation is enabled and the packet is an ARP packet, the system will check for ARP spoofing. If ARP spoofing is detected, a stream entry is added to the switch the attacker attended. Otherwise, the packet is forwarded. If Mitigation is not enabled, the packet is simply forwarded. Search includes a stream entry management component that removes expired stream entries. If the specified flow entry expires, it is removed from the system to maintain efficiency and ensure security measures are up to date. By implementing this proposed model, IoT devices in SDN networks can be protected against DDOS attacks and ARP spoofing, thus enhancing the overall network security and reliability.

**Keywords:** IOT, SDN, Mininet-WiFi, DDOS, MiTM, Machine Learning

## CHAPTER 1 INTRODUCTION

### 1.1 Introduction

There are many challenges and concerns regarding the security of the Internet of Things (IoT) [1], as these devices are uniquely identified in networks and are used in data analysis and decision making and can connect to the Internet. Currently, billions of devices and entities are connected to the Internet and share a IOT of information with organizations responsible for providing applications that produce data as these institutions can use this data to predict the applications and products desired by the consumer [2].

The applications of IoT devices are many, including simple home sensors, medical devices, cars, planes, and even nuclear reactors many IoT devices and applications operate without the use of any security mechanism. One of the main challenges facing the Internet of Things is how to provide security for the infrastructure that contains IoT devices [3] Because of the nature of the Internet of Things, communication may be devoid of confidentiality and authentication, which makes an attacker can easily target them by illegally intercepting and manipulating valuable sensor data in transit, by sending fraudulent requests or denial-of-service (DoS) attacks, taking over a physical device and turning it into a zombie to launch attacks on other systems. Exhaustion attacks are the most common attacks targeting the Internet of Things [4].

Networks use traditional methods of protection against electronic attacks such as firewalls and intrusion detection/ prevention systems at the edge of the network to prevent external electronic attacks. Due to the specific workings of these technologies, these defense mechanisms do not work directly with IoT networks [4]. Recent developments have created networks known as (SDN), and through this technology, the general behavior of the network is controlled by a central program, called the "SDN console". Which provides quick reactions to security threats that may occur, accurate filtering of data that passes through the network, and the appley of dynamic security policies [2].

SDN can automatically forward traffic when needed. Which greatly improves the performance of IoT applications. and better than Synchronization of virtual networks, storage and computing resources is provided and delivered immediately for data analysis. SDN promises to enable more management and security Communication, through the implementation concept of the programming networks [4].

In this study, we will investigate the use of SDN in the Internet of Things to improve network performance and enhance its security. We will build a proposed model for SDN with the Internet of Things that helps Mitigation man-in-the-middle attacks, DDOS attacks using best model in Machine Learning. Also, we will implement traffic separation using SDN techniques [5].

We will implement the proposed system using the Ubuntu 18.04 OS VM, Ryu SDN controller use python language, Mininet-WiFi to s simulate a public network containing IoT devices, regular computers, and network devices needed to build the target network, and use the OpenFlow protocol to implement SDN, and using Machine Learning to Selection best Model based on generating and analyzing the normal traffic of SDN network to Mitigation DDOS Attack. Our solution provides both confidentiality and integrity and mitigate various risks without the need to modify IoT devices [6].

## 1.2 Problem Statement

The Internet of Things faces many challenges and security problems resulting from a group of reasons, including the development of these devices with the necessary communication capabilities for the network, as these devices do not provide strong security. The number of such devices is increasing exponentially, and having single-tasking devices with rudimentary network connectivity increases the chances of launching attacks. The increase in the volume of data being generated through these devices makes it difficult to manage and direct this data. Also, these devices use insecure protocols like HTTP and Telnet protocol which makes man-in-the-middle, MITM attacks effective. Therefore, SDN technology will be used to reduce these issues and protect the proposed system from DDOS attacks, and man-in-the-middle attacks.

## 1.3 Research aim

The aim of this project is to build a network containing IoT devices, and to secure this network using the necessary security techniques, especially SDN technology, which in turn will be responsible for securing IoT devices, routing data as needed, and mitigation DDOS attacks, and man-in-the-middle (ARP Poisoning) attack, and train a model for detecting and mitigating DDoS attacks on SDN networks using Machine Learning.

## 1.4 Research objectives

- Build an SDN-based security model that has a built-in policy for securing IoT devices in the Wireless network and detecting and mitigating malicious ARP attacks .
- Train a model for detecting and mitigating DDoS attacks on Wireless networks using Machine Learning.
- Training different supervised ML algorithms on our own dataset that try to detect if the traffic is normal of a DDoS.
- Choosing the most appropriate algorithm that performs well on DDoS detection.
- Demonstration of the proposed model and the proposed security protocols using a real-world scenario involving IoT devices, showing how this model can protect the IoT infrastructure from attacks such as DDoS, and MiTM (ARP Poisoning).
- Analysis of the performance of the proposed system for securing IoT devices.

## 1.5 Scope and limitation of the project

This research focus on Improving the performance and protection of IoT devices, detecting and mitigating DDoS, MiTM attacks in Wireless networks.

## 1.6 Research Questions

Based on the problem statement mentioned above, the following questions are constructed:

- What is the gap and limitations that still exist in the previous literature studies related to this research?
- What are the techniques that are most suitable to this study, so that it can be used during the implementation stage?
- How can we evaluate the effectiveness and efficiency of the selected techniques?

- what are the tools and criteria that we are going to use in the evaluation stage?

## 1.7 Methodology

Mininet-WiFi simulator was used to create a software-defined networking (SDN) network consisting of one Ryu Controller, three OpenFlow switches, six access points, and twelve IoT devices. The network was simulated using a Ubuntu 18.04 OS VM.

To measure network parameters, tools provided by the simulator were utilized. Metrics such as delay, traffic in the topology, and traffic between hosts and IoT devices were analyzed using the sFlow-rt tool. The study also investigated security vulnerabilities by generating ARP poisoning and DDoS attacks using tools like Ettercap and Hping3.

To counter these attacks, for MiTM: the SDN-based security model used for detecting and mitigating MiTM (ARP poisoning) attacks, and for DDoS: machine learning algorithms were trained and applied in the Ryu Controller. The best algorithm for detecting and mitigating DDoS attacks were evaluated for their performance.

## 1.8 Organization of the project

This report consists of 5 chapters covering the mechanism used by the SDN to improve the performance and security of IoT devices and explain how to solve the problems of DDoS and MiTM Attacks. Here is an overview of the content of each chapter offered:

- **Chapter One:** This chapter provides a general introduction about the IoT and wireless Network. This chapter also discusses the problem statement, scope, objectives, and significance of this study.
- **Chapter Two:** This chapter covers the literature review and relevant information related to this study.
- **Chapter Three:** This chapter explains selected techniques used in this study in details. it also discusses the system model, system implementation, and DDoS attacks, and man-in-the-middle attacks on IoT devices in SDN.
- **Chapter Four:** This chapter discusses the system evaluation including the attacks results and performance evaluation.
- **Chapter Five:** This chapter introduces the conclusion, recommendations, and future works to improve this study.

## CHAPTER 2 LITERATURE REVIEW

### 2.1 Overview

In This chapter will explain some previous studies related to IOT have been applied to this domain. explain the definition, applications, architecture, and general security issues of Internet of Things (IoT). We then provide an overview of Software Defined Networking (SDN), with a description of the architecture that these networks consist of and the integration between it and the Internet of Things.

## 2.2 Related Work

Table 2-1 Related works on IOTs with SDN Technology

Year	Ref.	Method used
2016	[7]	Monitoring network traffic, Analyzing ARP requests, implementing mechanisms to validate the authenticity of ARP replies.
2017	[8]	softThings approach and machine learning in the IOT by applying the linear and nonlinear methods of the SVM algorithm in the SDN-controller.
2020	[9]	SDN with IoT networks using Raspberry Pi, Kodi Media Center, and OpenFlow Protocol.
2020	[10]	SDN security mechanisms using ONOS SDN Controller and Raspbian Virtual Machines and counteract malware packet injection, and DDoS attacks using Mirai.
2021	[11]	SDN security algorithm to detected MITM attack and block the attacker device using RYU SDN Controller and the pox controller.
2022	[12]	Detect a DDOS attack in the SDN controller using KNN and RF algorithms to block ports and drop packets that send from the source of the attack.
2023	[13]	Tested the KNN and DT algorithms in analyzing the traffic passing through the SDN-controller and classifying them into normal traffic and attack traffic, and then preparing the port that generates the attack traffic to protect the network from DDoS attack.
2023	[14]	Machine learning algorithms in software-defined networks (SDN) to achieve this goal, presents an overview of the key concepts, techniques, and challenges involved in using ML in SDN for DDoS detection and mitigation.

In 2016, Ahmed et al [7]. In this Paper the vulnerabilities present in the ARP protocol and proposes a mitigation technique to detect and prevent ARP spoofing attacks within an SDN environment. The proposed approach involves monitoring network traffic, analyzing ARP requests, and implementing mechanisms to validate the authenticity of ARP replies, thus enhancing the security of SDN against ARP spoofing attacks.

In 2017, Bhunia et al [9], They Proposed softThings approach that combines hierarchical SDN control planes and machine learning is proposed to detect and protect against DDOS attack in the IOT by applying the linear and nonlinear methods of the SVM algorithm in the SDN-controller on several scenarios, one of which is the IOT device. The target of the attack and the other in which the IOT device is the source of the attack, and the last scenario presents Two compromised IoT devices launch DDoS attack and the performance of the nonlinear SVM was the best in all scenarios.

In 2020, Abdullah et al [8]. They proposed a system model that integrates software-defined networking (SDN) with IoT networks to enhance security. Their approach involved addressing man-in-the-middle attacks against IoT devices that can only use HTTP, which is a difficult attack to defend against. To mitigate this attack, the authors presented a solution and implemented it using Raspberry Pi, Kodi Media Center, and the OpenFlow Protocol. The proposed technique was evaluated, and the results showed that it was more resilient to cyber-attacks, and presents a promising approach to enhancing the security of IoT networks using SDN and addressing critical attacks such as man-in-the-middle attacks.

In 2020, Kallol et al [10]. They proposed a private security architecture for networks that have authenticated IoT devices and precise policies for securing flows in the IoT infrastructure. They achieved authentication using a lightweight IoT device authentication protocol and implemented and validated the proposed security mechanisms using ONOS SDN Controller and Raspbian Virtual Machines. And demonstrated how the proposed security mechanisms can effectively counteract malware packet injection and DDoS attacks using Mirai. Thus, their proposed architecture provides a promising solution for enhancing the security of IoT networks, particularly in countering the growing threat of DDoS attacks.

In 2021, Saritakumar et al [11], In this proposed model, the MITM attack was detected by an algorithm based on IP-MAC addresses bindings in the SDN-controller. This algorithm was applied to the RYU Controller and the pox controller. It was concluded that the pox controller consumes more time to process data packets during the attack, but it consumes the CPU less than RYU Controller.

In 2022, Mohsin et al [12], In this paper, KNN and RF algorithms were applied to detect a DDOS attack in the SDN controller, where these algorithms work on block ports and drop packets for the source of the attack. These algorithms were applied to several topologies, which are single, linear, and multi controllers. The two algorithms display the best performance and accuracy in this work.

In 2023, Jean et al [13], In this paper, the researchers tested the KNN and DT algorithms in analyzing the traffic passing through the SDN-controller and classifying them into normal traffic and attack traffic, and then preparing the port that generates the attack traffic to protect the network from DDoS attack, and the results were high accuracy of DT, which equals 99.77% compared to KNN, which equals 99.38%.

In 2023, Naman et al [14], This paper discusses the growing threat of DDoS attacks and the need for effective detection and mitigation techniques. They Proposed using machine learning algorithms in software-defined networks (SDN) to achieve this goal, presents an overview of the key concepts, techniques, and challenges involved in using ML in SDN for DDoS detection and mitigation. The evaluated six ML algorithms, including Logistic Regression, Naive Bayes, K-Nearest Neighbor, Support Vector Machine, Decision Tree, and Random Forest, and found that KNN, DT, and RF were the most effective. The evaluation results showed that the proposed model was able to recognize DDoS attacks accurately and quickly.

### 2.3 Internet of Things (IOT)

The Internet of Things refers to billions of physical devices in the world that are now connected to the Internet, and all devices collect and share data. Thanks to the arrival of very cheap computer chips and the ubiquity of wireless networks, it is possible or possible to turn anything into a part of the Internet of Things. And so, connecting all these different objects and adding sensors to them adds a level of digital

intelligence to devices that would otherwise be dumb, enabling them to communicate data in real time without involving any human beings. The Internet of Things makes the fabric of the world around us smarter and more responsive, merging the physical and digital worlds [15].

## 2.4 Applications of Internet of Things (IoT)

**1. Smart Homes:** Smart homes are one of the best and most practical applications in the field of Internet of Things, as they move comfort and home safely to the next level. Although different levels appear in which the Internet of Things is applied to smart homes, the appropriate level is the level that mixes smart amenities and entertainment systems together. For example, the electricity meter with the Internet of Things gives knowledge of the daily use of water, automatic lighting systems and advanced locking, as well as connected surveillance systems and all of them are proportional to the concept of smart homes by the Internet of Things and its development, we are sure that the vast majority of devices will be smarter, which allows the home to improve optimized safety.



*Figure 2-1 Smart Home in IOT*

**2. Smart City:** Not only access to the Internet for people in the city but for existing devices so that connected technology is integrated into infrastructure requirements and some vital concerns such as waste management, traffic, electricity, water distribution and more. All these actions are aimed at eliminating some of the day-to-day challenges and bringing more comfort.

*Figure 2-2 Smart City in IOT*



**3. Self-driven Cars:** Self-driving cars We've seen a lot about them, Google tried them, Tesla tested them but they were later shelved. This is because we are dealing with human lives, so we need to ensure that technology has everything necessary for guarantee and safety, and self-driving cars use many built-in systems and sensors that are connected in the cloud and the Internet to generate data and send it to them to make informed decisions through machine learning. Also, although it will take more years for



the technology to fully develop and countries to adjust policies and laws, what we are currently witnessing are the best applications of the Internet of Things.



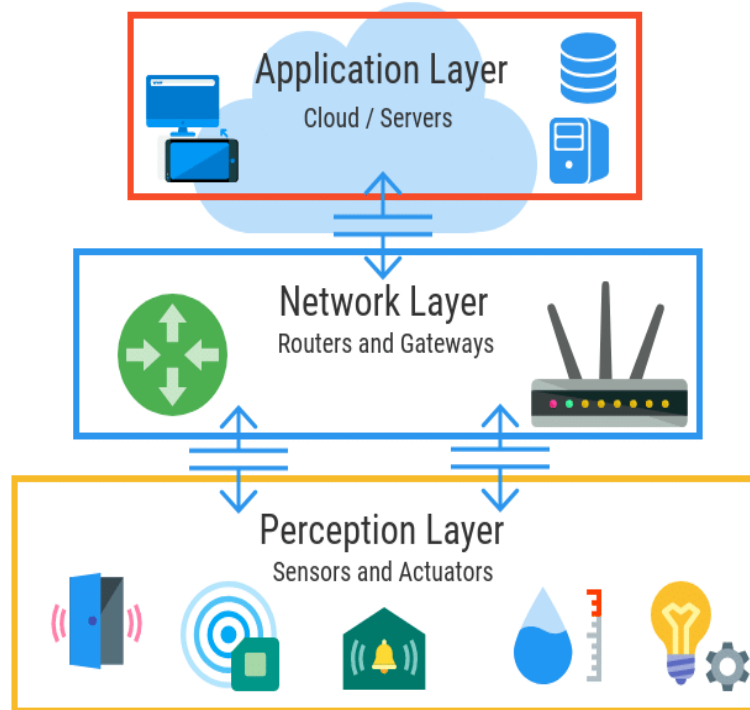
*Figure 2-3 Self Driven car in IOT*

## 2.5 Architecture of Internet of Things (IoT)

IoT architecture consists of network structure, devices, and cloud technology that allows IoT devices to communicate with each other. IoT architecture consists of three layers:

- Perception Layer.
- Network Layer.
- Application Layer.

These layers support the Internet of Things devices by collecting and processing data. Consequently, this structure exceeds the OSI model to include converting data into information that is usable or usable. These ideas allow companies to take immediate action through the use of automation, machine learning and artificial intelligence [16].



*Figure 2-4 Layer of Internet of Things*

## 2.6 Software Defined Networking (SDN)

SDN presents an appealing alternative to traditional networks by offering a software-based, programmable infrastructure that enhances speed, flexibility, and adaptability. The centralized control provided by SDN controllers enables efficient management and customization of packet flow, ultimately enhancing network performance and responsiveness. [17].

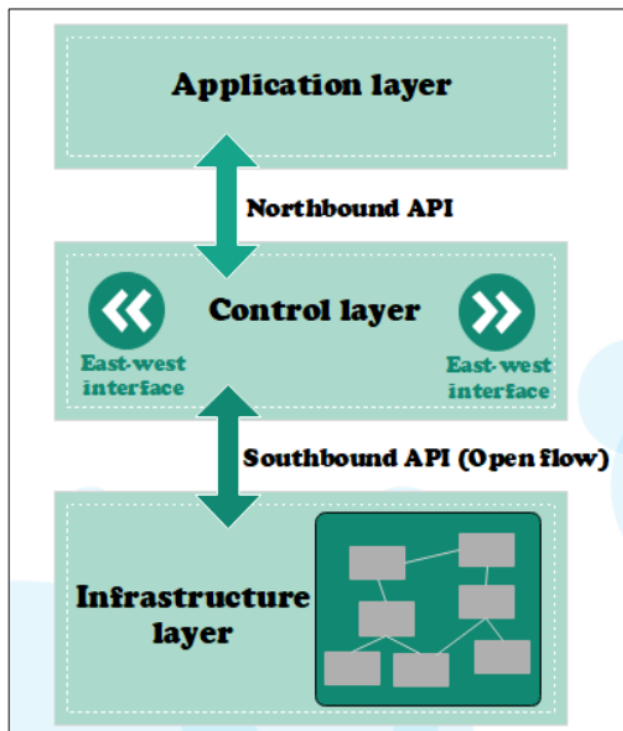
## 2.7 SDN Architecture

SDN architecture characterizes how a networking and registering framework can be fabricated utilizing a blend of open, software based on technologies and item networking equipment, six fundamental components make up a SDN design [18].

- **The management plane:** Sometimes called an application layer. It is a collection of network application that manage the control of SDN. SDN-enabled networks, instead of utilizing a common line interface, depend on programmability. Carrying out new applications and services, for example, routing, load balancing, strategy implementation, requires flexibility and ease. It additionally empowers for network arrangement and automation utilizing current APIs.
- **The control plane:** The most complex and smart part of a SDN design. It has at least one controller that utilization the southbound interface to send different kinds of rules and policies to the infrastructure layer.
- **The data plane:** Sometimes called to an infrastructure layer, addresses the organization's sending devices (switches, routers, load balancers, and so forth) It communicates with the

control plane by means of southbound APIs, getting sending rules and policies and applying them to the suitable devices.

- **The northbound interfaces:** That permit correspondence between the control and the management layers are basically open-source application programming interfaces (APIs).
- **The east-west interfaces:** Permit a few controllers to convey. They utilize a warning and informing framework or a circulated directing convention like BGP or OSPF to communicate.
- **The southbound interfaces:** Give association between the control plane and the data plane, which are protocols that permit the controller to communicate rules to the data plane. For SDN-



enabled networks, the OpenFlow protocol is the most regularly recognized and carried out southbound API. The Open Networking Foundation (ONF) [19], which is supported by IT industry titans like Facebook, Cisco, Google and others, has normalized OpenFlow.

*Figure 2-5 Layer of Software Defined Network*

## 2.8 SDN VULNERABILITIES

SDN, like any other technology is vulnerable to security threats that can lead to severe consequences. Here are the four main categories of security threats in SDN:

1. **Threats against an SDN Controller:** The SDN Controller is the brain of the network, and any attack on it can lead to the failure of the entire network. The Controller can be targeted with various attacks, such as denial-of-service (DoS), remote code execution, and SQL injection.
2. **Threats against networking devices:** Networking devices such as switches, routers, and firewalls can be compromised to launch attacks. Attackers can exploit vulnerabilities in the firmware or software of these devices to gain unauthorized access or launch DoS attacks.

3. **Threats against communications between the Controller and the networking devices:** The communication between the Controller and networking devices can be attacked by exploiting vulnerabilities in the communication protocols, such as OpenFlow. Attackers can intercept and modify traffic, inject malicious packets, or even impersonate the Controller.
4. **Threats against communications between different SDN Controllers in different domains:** SDN enables the creation of a distributed network with multiple Controllers, which can communicate with each other. However, this communication can be vulnerable to various attacks such as man-in-the-middle (MITM), eavesdropping, and data tampering [8].

## 2.9 SDN ATTACKS

### 2.9.1 ARP spoofing Attack

ARP spoofing attack, also known as ARP poisoning, is a type of cyber-attack that exploits vulnerabilities in the Address Resolution Protocol (ARP) to intercept and modify network traffic. The attack involves the attacker sending fake ARP messages to the devices on the network, which results in the ARP cache being poisoned with the attacker's MAC address associated with a legitimate IP address. Once the ARP cache has been poisoned, the attacker can intercept and modify network traffic, which can result in sensitive information being compromised or further attacks being launched.

ARP spoofing attacks are a significant concern for network security, as they can be difficult to detect and can result in serious consequences. To prevent ARP spoofing attacks, various security measures such as using static ARP tables, ARP spoofing detection software, network segmentation, and encryption and authentication protocols have been proposed. These measures aim to enhance the security of networks and prevent unauthorized access and data breaches. Overall, ARP spoofing attacks represent a significant threat to network security and require appropriate measures to be implemented to mitigate their risks [20].

### 2.9.2 DDoS Attacks

Distributed Denial of Service (DDoS) attacks have become a critical issue in network security research. This type of cyber-attack involves a coordinated effort to overload a targeted network with traffic from multiple sources, effectively rendering it inaccessible to legitimate users [21].

DDoS attacks can be launched using various methods, such as flooding the network with traffic, exploiting vulnerabilities in network protocols or devices, or by using botnets consisting of compromised devices. The impact of these attacks can be severe, with financial, operational, and reputational damages to the targeted organizations.

To mitigate the risks of DDoS attacks, researchers have proposed several defense mechanisms, including anomaly detection and mitigation, rate limiting, network-based mitigation, and cloud-based mitigation. These mechanisms aim to detect and prevent DDoS attacks, as well as minimize the impact of successful attacks.

Further research is needed to develop more advanced defense mechanisms and to improve the understanding of DDoS attack methods and characteristics. This can help organizations to effectively protect their networks against these attacks and maintain the availability and integrity of their services [21].

## 2.10 RYU CONTROLLER

SDN (Software-Defined Networking) is a networking approach that separates the control plane from the data plane in network devices such as switches and routers. The control plane is moved to a central controller, which can be programmed to manage the network traffic dynamically. Ryu is a popular Python-based SDN controller that provides a flexible and customizable platform for developing and testing SDN applications.

Ryu Controller is another widely used Python-based SDN controller that provides a flexible and programmable platform for developing and deploying SDN applications. It is designed to support various OpenFlow versions, including OpenFlow 1.0, 1.1, 1.2, 1.3, and 1.4, ensuring compatibility with a wide range of switches and controllers [22].

Ryu is built as a modular framework, allowing developers to customize and extend its functionality to build complex SDN applications. It adopts an event-driven programming model, enabling rapid development and testing of SDN applications. Ryu facilitates the creation of reactive applications that can promptly respond to changes in the network environment [22].

Ryu can also be integrated with other networking tools and technologies, such as Mininet and Wireshark, to facilitate testing and debugging of SDN applications in diverse network environments. As an open-source project, Ryu benefits from a large community of developers and users who contribute to its development, provide support, and share resources, ensuring a vibrant ecosystem for SDN developers.

Generally, Ryu Controller is a powerful and versatile SDN controller that offers comprehensive support for multiple OpenFlow versions, an event-driven programming model, and seamless integration with complementary networking tools. These features make Ryu a favored choice among SDN developers and a valuable tool for scientific research and experimentation in the field of SDN and related networking technologies. [23].

## 2.11 Machine Learning

Machine Learning is a main branch of artificial intelligence as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty, and which focuses mainly on machine learning from their experience and making predictions based on its experience. IT consists of algorithms that build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task [24].

Machine learning algorithms are divided into 4 types according to whether the training data are labeled or not and according to the training supervision they have received. These types are supervised learning, unsupervised learning, semisupervised learning and reinforcement learning [24].

- **Supervised learning:** In this method, algorithms (such as Linear Regression, Logistic Regression, Decision Trees, Naïve Bayes Classification and Random Forest.) learn from labeled and predefined data. After understanding the training data, it determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data. Two

groups or categories of algorithms come under the umbrella of supervised learning [25]. They are Regression and Classification.

- **Unsupervised learning:** In this method, algorithms (such as K-means, Hierarchical clustering, Fuzzy C-Means) do not learn from labeled data. Instead, it tries work on its own to discover information, this is sometimes called knowledge discovery [25]. These algorithms combine similar data into groups, and then the test sample is classified based on these groups.
- **Semi-supervised learning:** this method combined supervised learning and unsupervised learning [26]. it uses labeled data for training – typically a small amount of labeled data with a large amount of unlabeled data.
- **Reinforcement Learning:** this method is very different from the other methods as a basic principle. in this method, the algorithms (such as Q-Learning, Temporal Difference (TD) and Deep Adversarial Networks) receive feedback for the decision that are made so, it builds its own rule [26]. it is one of the types of learning without supervision, in which the machine interacts with the environment and adopt its experiences based on this interaction.

### 2.11.1 Relation Machine Learning in SDN Network

Machine learning in SDN is distributed feature of traditional networks, machine learning techniques are hard to be applied and deployed to control and operate networks. Software defined networking (SDN) brings us new chances to provide intelligence inside the networks. The capabilities of SDN (e.g., logically centralized control, global view of the network, software-based traffic analysis, and dynamic updating of forwarding rules) make it easier to apply machine learning techniques. In this research, we provide a comprehensive survey on the literature involving machine learning algorithms applied to SDN. First, the related works and background knowledge are introduced. Then, we present an overview of machine learning algorithms. In addition, we review how machine learning algorithms are applied in the realm of SDN, from the perspective of traffic classification, routing optimization, quality of service/quality of experience prediction, resource management and security.

In this study, supervised learning methods will be used to take advantage of having a manually-labeled good dataset. machine learning algorithms used in the implementation phase are: Naive Bayes, Random Forest, Decision tree, K Nearest Neighbor and Logistic classification [27] these algorithms are examined in detail as follow:

#### - Naive Bayes:

Naive Bayes is a classification technique based on Bayes' theorem with an assumption of independence between predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Bayes theorem provides a way of calculating posterior probability  $P(A|B)$  from  $P(A)$ ,  $P(B)$  and  $P(B|A)$ . Look at the equation below:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (2-1)$$

Where:  $P(A/B)$ : the conditional probability that event A occurs, given that B has occurred. This is also known as the posterior probability.

$P(A)$  and  $P(B)$ : probability of A and B without regard of each other.

$P(B/A)$ : the conditional probability that event B occurs, given that A has occurred. **advantages of the naive Bayes classifier are:**

- The training time is short.
- the computational cost is very low.

**Disadvantage of the naive Bayes classifier is:**

The assumption of independence among feature is clearly almost always wrong and for this reason naive Bayes classifiers are usually less accurate than other Machine learning algorithms.

- **Decision Tree:**

A Decision tree is a supervised learning method used for classification and regression. classifier expressed as a recursive partition of the instance space; the goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features [27].

The Decision tree consists of nodes that form a Rooted Tree, meaning it is a Directed Tree with a node called root that has no incoming edges. All other nodes have exactly one incoming edge. A node with outgoing edges is called internal node or test nodes. All other nodes are called leaves (also known as terminal nodes or decision nodes).

- **Random forest:**

Random forest [27] is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithms of the same type i.e., multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

**Advantages of using Random Forest:**

- This algorithm is very stable.
- This algorithm can work well even the dataset is very large and complicated.
- The random forest algorithm also works well when data has missing values or it has not been scaled well. It replaces these lost values with their own created values.

**Also, this algorithm has a disadvantage:**

A major disadvantage of random forests lies in their complexity. They required much more computational resources, owing to the large number of decision trees joined together. And due to their complexity, they require much more time to train than other comparable algorithms.

- **K Nearest Neighbor:**

The K Nearest Neighbors algorithm [27] is an instance-based. It has been considered as one of the simplest of all machine learning algorithms. This algorithm depends on the assumption that the instances within a dataset will exist in close proximity to other instances that have 17 similar properties. KNN determines the class of new instances that is unclassified by observing the class of its nearest neighbors,

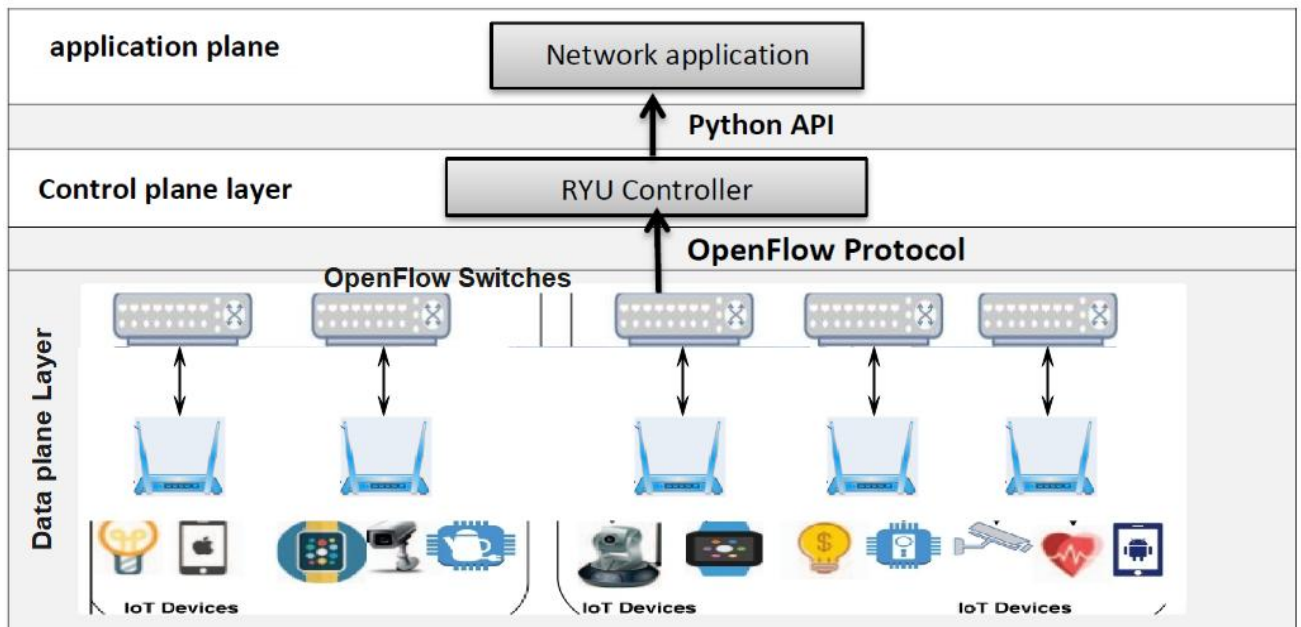
which tagged with a classification label. The KNN locates the k nearest instances to the query instance and determines its class by identifying the single most repeat class label.

## CHAPTER 3 METHODOLOGY

### 3.1 Overview

In This chapter discussed the methods and implementation that have been used in this project and evaluate the proposed SDN-based security model for IoT devices. The chapter is divided into several sections that describe the steps taken to carry out the research objectives.

### 3.2 Design of SDN Network



*Figure 3-1 SDN Network framework*

In the figure 3-1 SDN framework's data plane, multiple node/hosts are virtually created using Mininet-WiFi [28]. These nodes/hosts are connected to an OpenFlow switch, which implements the SDN protocols. The OpenFlow protocol facilitates communication between the data plane and the control plane of the SDN framework.



The control plane is responsible for managing and controlling the data plane and the switches. It defines rules for network behavior and also monitors the flow of network traffic. In this particular setup, the Ryu [22] controller is utilized as the controller for the SDN framework. Ryu provides programming capabilities and enables us to control routing operations within the network.

The control plane is programmed using the Python language since Ryu is a Python-based controller. It uses a Python-based API to communicate with the application layer, which in this case, consists of network traffic applications.

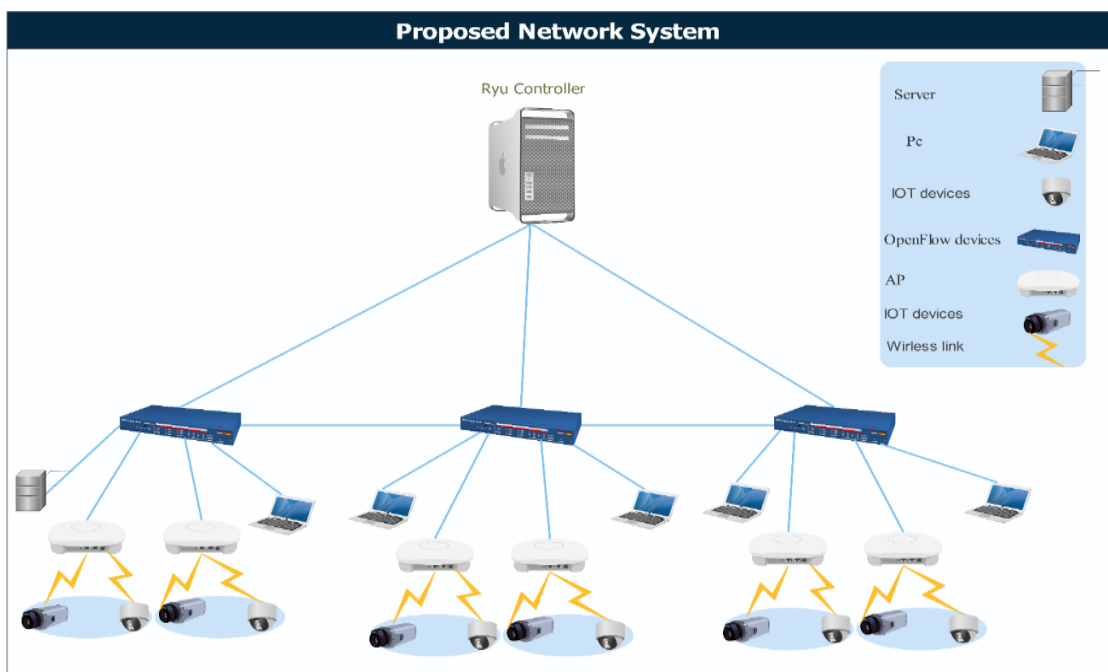
### 3.3 Proposed System

The proposed system model aims to address the security challenges and concerns related to the Internet of Things (IoT) devices. With billions of devices and entities connected to the Internet, sharing a lot of information, it is critical to provide security for the infrastructure that contains IoT devices.

The traditional methods of protection against electronic attacks, such as firewalls and intrusion detection/prevention systems, do not work directly with IoT networks. To overcome this challenge, recent developments have created networks known as Software-Defined Networking (SDN), which use a central program called the "SDN console" to control the general behavior of the network.

The proposed model will investigate the use of SDN in IoT to enhance network performance and security. The model will include a proposed SDN system that helps prevent man-in-the-middle attacks, DDOS attacks Using Machine Learning, and implements traffic separation using SDN techniques.

The proposed system will be implemented using Ubuntu 18.04 OS VM, SDN Ryu [22] controller use Python language, Mininet-WiFi [28] to simulate a public network containing IoT devices, regular computers, and access point devices needed to build the target network, and the OpenFlow protocol to implement SDN.



### Figure 3-2 Proposed Network System

## 3.4 Methodology and implementation: DDOS Attack

### 3.4.1 Generating Dataset from Topology Network

The first process is creating the dataset. at first, will run Ryu [22] controller in the control plan and run network topology by Mininet-WiFi [28] in the data plane. in the data plane when run script normal on topology network, will collect traffics by the RYU controller and check from this collected traffics, if traffic was normal will run script normal and store traffic information in the normal file dataset. and after collected normal traffics will stop run script normal in the data plane and run script DDOS on topology will collect DDoS traffic in the control plane by the RYU controller and check from this collected traffic information, and if was DDOS, will run script DDOS and store this traffics in the DDoS file dataset. After created two files contain on normal and DDoS traffics. will merge two files in the one file dataset by the RYU controller for the ML algorithm to use. Figure 3-3 shows a simple diagram for generating dataset.

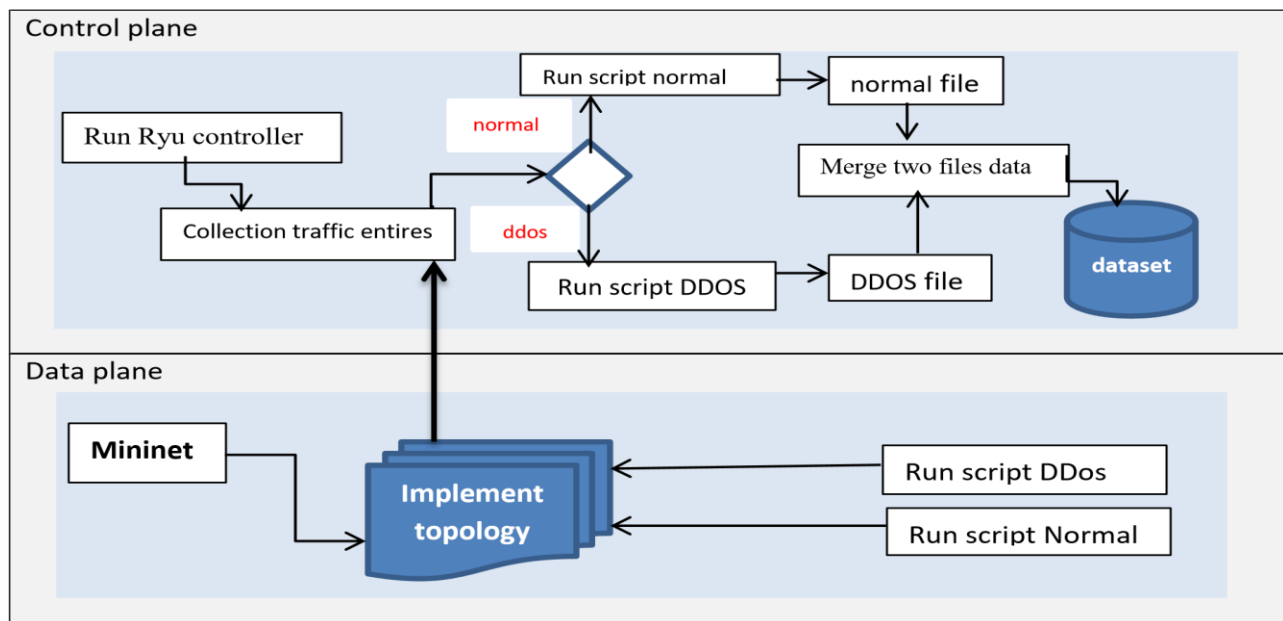


Figure 3-3 Generating Dataset for Project

The dataset creation process involves running the RYU [22] controller and setting up the network topology using Mininet-WiFi [28]. Normal and DDoS traffic is collected and stored in separate CSV files, which are then merged into a single dataset file for machine learning analysis. Thus, the data is collected considering all 22 parameters of feature extraction different in the methodology namely as Table 3-1.

**Table 3-1: Feature Extraction of Dataset**

No	Feature Name	Description
1	Timestamp	which is record of date and time for each active flow entry state at Ryu controller time request.
2	datapath_id	which is the identity number of active OpenFlow switch in data plane layer.
3	flow_id	The unique identifier for the flow that the OpenFlow message pertains to.
4	ip_src	IP source which is ipv4 address source of active flow entry.
5	tp_src	TCP and UDP IP source which can be the active communication port of entry source.
6	ip_dst	IP destination which is ipv4 address destination of active flow entry
7	tp_dst	TCP and UDP IP destination which can be the active communication port of entry destination.
8	ip_proto	IP Protocol which is identifying type of active flow entry protocol as ICMP, TCP, UDP.
9	ICMP_code	which is define kind of ICMP protocol message.
10	ICMP_type	which is one of ICMP protocol message.
11	flow_duration_sec	which is the used time of active flow entry, in seconds and nanoseconds.
12	flow_duration_nsec	
13	idle_timeout	The duration in seconds after which the flow entry will be deleted if it is not referenced.
14	hard_timeout	The duration in seconds after which the flow entry will be deleted regardless of whether it is referenced or not.
15	Flags	A flag indicating the type of operation to be performed.
16	packet_count	The number of packets associated with the flow.
17	byte_count	The number of bytes associated with the flow.
18	packet_count_per_second	which are the total packets count of active flow entry divided by its flow duration in second and nano seconds.
19	packet_count_per_nsecond	
20	byte_count_per_second	which are the total bytes count of active flow entry divided by its flow duration in second and nano seconds.
21	byte_count_per_nsecond	
22	label	

### 3.4.2 Dataset Scaling

After getting features collected from flow state statistics of SDN network by creating legitimate traffics and DDoS traffics then collected features from flow state statistics of these traffics stored in csv file as

dataset. The dataset it includes input as features (columns) which is 22 features and 2667523 rows as instances. and output which is label 0 for normal traffics and 1 for DDOS traffics,

### 3.4.3 Features selection

It was important to select important features from un important feature in the dataset before applying it to machine learning model for getting good performance and classification results of models also high accuracy with no high load to controller to classify normal traffics from DDOS attack traffics because more features make high load to controller but high accuracy and good prediction of Machine learning models so the tradeoff between these reasons must be done ,the important features were selected by using CLASSIFIER MODEL .

In this context, when we determine the attribute importance to define the attack, we are choosing the more generic and invariant attributes. Because much more information about whether it is an attack or not will be given by the shape of the data. The distribution of features and 17 attributes with the most significance value DDOS attack illustrated in Figure 3-3-b and Table 3-2.

*Figure 3-3 Accuracy of Important Features*

Feature	Accuracy
0	0.901467
1	0.999733
2	0.912267
3	0.9924
4	0.912267
5	0.596
6	0.5672
7	0.8388
8	0.852533
9	0.7108
10	0.967867
11	0.968133
12	0.959333
13	0.959333
14	0.959333
15	0.959333

*Table 3-2 Important Features of Dataset*

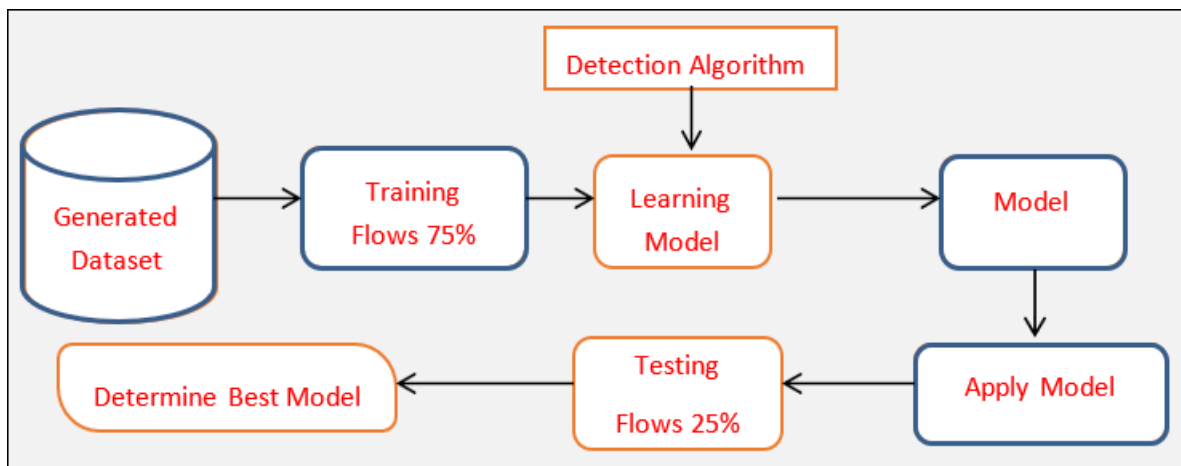
No	Feature Name	No	Feature Name
1	datapath_id	10	flow_duration_nsec
2	ip_src		
3	tp_src	11	packet_count
4	ip_dst	12	byte_count
5	tp_dst	13	packet_count_per_second

6	ip_proto	14	packet_count_per_nsecond
7	icmp_code	15	byte_count_per_second
8	icmp_type	16	byte_count_per_nsecond
9	flow_duration_sec	17	label

### 3.4.4 Creation of Training and Testing Data

In the machine learning process, data is essential for training, testing, and evaluating the algorithm's performance. so that, the DDoS Attack TrafficFlow dataset used in this application didn't come pre-divided into training and testing sets. To address this, the data needed to be split into two parts: a training set for the algorithm to learn from, and a testing set to assess its performance. The sklearn command "train\_test\_split" was used to achieve this, randomly dividing the data according to user-specified sizes. In this case, the commonly preferred partitioning ratio of 25% for testing and 75% for training was adopted. To ensure reliable results, the process of creating training and testing data was repeated 10 times consecutively. The final results are obtained by calculating the average of these repeated operations, providing a more robust assessment of the algorithm's performance.

### 3.4.5 Implement ML Algorithms



*Figure 3-4 Implement ML algorithm*

In this work, 6 different machine learning algorithms (LR, NB, SVM, KNN, DT, RF) were used for the implementation process. These algorithms were applied to the attack file using the attributes obtained in the feature selection phase.

To ensure reliable results, each algorithm was repeated 10 times. This repetition helps eliminate irregular values and provides a more robust evaluation. After the repetitions, the average values were calculated to determine the results, In Fig 3-4. illustrates that all the models were trained and tested using the applied dataset, considering different numbers and types of features. This approach allows for a fair comparison between the models, as they are evaluated on datasets with varying feature sets [26].

### 3.4.6 Applying the Best Model

The machine learning classification models used in this project were moderated, and the best model was characterized by high accuracy and high performance, after models trained on training data with different numbers and types of features and tested on test data, and rating scales used in evaluation. These models and compare evaluation results for each model such as confounding matrix, accuracy, degree of drag and stress as well as learning and time prediction. The best selected model was run in Ryu console as application to test this model in real time and network. which had normal smuggling and attacking by working on hosts and Ryu console to test this model if the prediction is good or not, then solve the problem of the attack occurring in SDN network with its detection and mitigation by this proposed model [26].

### 3.4.7 DDOS Attack

A DDOS (Distributed Denial of Service) attack is an attempt by a hostile agent to block normal network traffic to a service, an entire network, a server, etc., by flooding the victim with abnormal Internet traffic. Compared to a DOS (Denial of Service) attack, where the malicious source is a single network host, DDOS attacks exploit multiple compromised computer systems as attack sources [21].

Among the different types of DDOS attacks existing in reference to the ISO/OSI stack, we chose to implement one of the protocol-based attacks, also known as state-exhaustion attacks, which lead to a service interruption through excessive resource consumption. In particular, in our project implementation, the chosen victim was one of the hosts IoT Device in the network. However, if there are no forwarding rules in the switch, packets pass through the SDN controller first, so an attack of this kind can not only exhaust the resources of the victim but also slow down the functions performed by the controller, which has to manage a huge number of abnormal packets.

Using the hping3 tool to generate a high volume of ICMP (Ping) packets targeting a specific IOT device IP address with a destination port of 80. The goal is to overwhelm the IOT resources by flooding it with ICMP packets, causing network congestion and disrupting its normal operation.

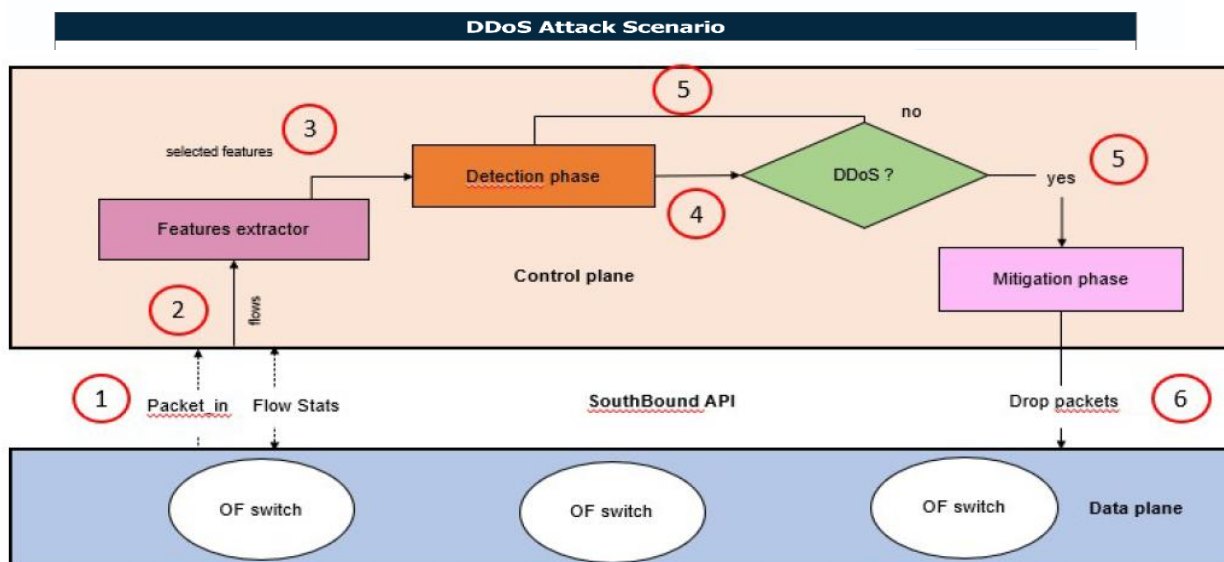
### 3.4.8 The Proposed Algorithm for Detecting and Mitigating DDOS

After the data collection and setting the controller to the detection state, the system employs a decision tree algorithm to predict whether incoming traffic is normal or an attack. During the generation of normal traffic, the algorithm correctly identifies it as normal and allows it to flow through. However, when attack traffic is generated, it promptly recognizes it as a DDoS attack and takes immediate action.

Upon detection of the attack traffic, the system identifies the type of attack and proceeds to block the port through which the malicious traffic is entering. This port-blocking mechanism is activated for a fixed duration of 120 seconds, after which the port is automatically unblocked. However, if the attack persists, the system continues to detect and block the port for additional 120-second intervals. Throughout this process, normal traffic from other ports is allowed to flow freely within the network. Thus, cycle of detection, blocking, and unblocking is repeated for as long as the attack remains active, ensuring that the network remains protected from the ongoing malicious activity. Refer to Fig. 3-5 for a visual representation of the detection and mitigation of DDOS Using Machine Learning.

**Figure 3-5 Algorithm for Detecting and Mitigating DDOS**

### 3.4.9 Implementation: First Scenario #: DDOS Attack



**Figure 3-6 Network Topology for DDOS**

In the given scenario, in Fig. 3-6 the attackers, H2 and H5, by hping3 tool generate a high volume of ICMP (Ping) packets targeting a specific IOT device IP address (10.0.0.8) with a destination port of 80. The goal is to overwhelm the IOT resources by flooding it with ICMP packets, causing network congestion and disrupting its normal operation.

The attackers use the following command to initiate the ICMP flood attack:

### hping3 -I -V -d 120 -w 64 -p 80 --rand-source --flood 10.0.0.8

```

eng@CND:~$ hping3 -I -V -d 120 -w 64 -p 80 --rand-source --flood 10.0.0.8
64 bytes from 10.0.0.5: icmp_seq=16 ttl=64 time=5.12 ms
64 bytes from 10.0.0.5: icmp_seq=17 ttl=64 time=5.25 ms
64 bytes from 10.0.0.5: icmp_seq=18 ttl=64 time=1.36 ms
64 bytes from 10.0.0.5: icmp_seq=19 ttl=64 time=2.21 ms
64 bytes from 10.0.0.5: icmp_seq=20 ttl=64 time=3.10 ms
64 bytes from 10.0.0.5: icmp_seq=21 ttl=64 time=2.26 ms
64 bytes from 10.0.0.5: icmp_seq=22 ttl=64 time=1.94 ms
64 bytes from 10.0.0.5: icmp_seq=23 ttl=64 time=2.73 ms
64 bytes from 10.0.0.5: icmp_seq=24 ttl=64 time=2.36 ms
64 bytes from 10.0.0.5: icmp_seq=25 ttl=64 time=2.32 ms
^C
--- 10.0.0.5 ping statistics ---
25 packets transmitted, 25 received, 0% packet loss, time 2412ms
rtt min/avg/max/ndev = 1.215/6.113/32.551/8.178 ms
root@CND:~/mininet-wifi# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=10.0 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=15.3 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=2.12 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=1.14 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=1.20 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=1.06 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=2.07 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=5.09 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=13.9 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=16.3 ms
64 bytes from 10.0.0.1: icmp_seq=11 ttl=64 time=3.44 ms
64 bytes from 10.0.0.1: icmp_seq=12 ttl=64 time=15.0 ms
^C
--- 10.0.0.1 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11032ms
rtt min/avg/max/ndev = 1.068/8.236/16.593/5.627 ms
root@CND:~/mininet-wifi# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=13 ttl=64 time=5.24 ms
64 bytes from 10.0.0.1: icmp_seq=14 ttl=64 time=5.21 ms
64 bytes from 10.0.0.1: icmp_seq=15 ttl=64 time=4.88 ms
64 bytes from 10.0.0.1: icmp_seq=16 ttl=64 time=5.26 ms
64 bytes from 10.0.0.1: icmp_seq=17 ttl=64 time=5.11 ms
64 bytes from 10.0.0.1: icmp_seq=18 ttl=64 time=5.11 ms
64 bytes from 10.0.0.1: icmp_seq=19 ttl=64 time=5.21 ms
64 bytes from 10.0.0.1: icmp_seq=20 ttl=64 time=5.09 ms
64 bytes from 10.0.0.1: icmp_seq=21 ttl=64 time=4.90 ms
64 bytes from 10.0.0.1: icmp_seq=22 ttl=64 time=5.06 ms
64 bytes from 10.0.0.1: icmp_seq=23 ttl=64 time=5.11 ms
64 bytes from 10.0.0.1: icmp_seq=24 ttl=64 time=4.57 ms
64 bytes from 10.0.0.1: icmp_seq=25 ttl=64 time=4.38 ms
64 bytes from 10.0.0.1: icmp_seq=26 ttl=64 time=5.34 ms
64 bytes from 10.0.0.1: icmp_seq=27 ttl=64 time=5.55 ms
64 bytes from 10.0.0.1: icmp_seq=28 ttl=64 time=4.43 ms
64 bytes from 10.0.0.1: icmp_seq=29 ttl=64 time=5.26 ms
64 bytes from 10.0.0.1: icmp_seq=30 ttl=64 time=5.37 ms
64 bytes from 10.0.0.1: icmp_seq=31 ttl=64 time=5.22 ms
64 bytes from 10.0.0.1: icmp_seq=32 ttl=64 time=5.22 ms
64 bytes from 10.0.0.1: icmp_seq=33 ttl=64 time=4.65 ms
64 bytes from 10.0.0.1: icmp_seq=34 ttl=64 time=10.10 ms
64 bytes from 10.0.0.1: icmp_seq=35 ttl=64 time=5.22 ms
64 bytes from 10.0.0.1: icmp_seq=36 ttl=64 time=4.09 ms
64 bytes from 10.0.0.1: icmp_seq=37 ttl=64 time=4.36 ms
64 bytes from 10.0.0.1: icmp_seq=38 ttl=64 time=4.94 ms
64 bytes from 10.0.0.1: icmp_seq=39 ttl=64 time=6.24 ms
64 bytes from 10.0.0.1: icmp_seq=40 ttl=64 time=4.97 ms
64 bytes from 10.0.0.1: icmp_seq=41 ttl=64 time=5.38 ms
64 bytes from 10.0.0.1: icmp_seq=42 ttl=64 time=5.03 ms
64 bytes from 10.0.0.1: icmp_seq=43 ttl=64 time=5.12 ms
^C
--- 10.0.0.1 ping statistics ---
43 packets transmitted, 42 received, 2% packet loss, time 42184ms
rtt min/avg/max/ndev = 4.576/6.340/20.220/3.656 ms
root@CND:~/mininet-wifi#
    
```

Figure 3-7 Normal Traffic In Controller

```

eng@CND:~$ hping3 -I -V -d 120 -w 64 -p 80 --rand-source --flood 10.0.0.8
using h2-eth0, addr: 10.0.0.5, MTU: 1500
HPING 10.0.0.8 (h2-eth0 10.0.0.8): icmp mode set, 28 headers + 120 data bytes
hping in flood mode, no replies will be shown

--- 10.0.0.8 hping statistic ---
2485707 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@CND:~/mininet-wifi# ping 10.0.0.11
PING 10.0.0.11 (10.0.0.11) 56(84) bytes of data:
From 10.0.0.5 icmp_seq=1 Destination Host Unreachable
From 10.0.0.5 icmp_seq=2 Destination Host Unreachable
From 10.0.0.5 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.0.11 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4094ms
pipe 4
root@CND:~/mininet-wifi# ping 10.0.0.14
PING 10.0.0.14 (10.0.0.14) 56(84) bytes of data:
From 10.0.0.5 icmp_seq=1 Destination Host Unreachable
From 10.0.0.5 icmp_seq=2 Destination Host Unreachable
From 10.0.0.5 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.0.14 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4097ms
pipe 4
root@CND:~/mininet-wifi# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
From 10.0.0.5 icmp_seq=1 Destination Host Unreachable
From 10.0.0.5 icmp_seq=2 Destination Host Unreachable
From 10.0.0.5 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 0 received, +3 errors, 100% packet loss, time 3079ms
pipe 4
root@CND:~/mininet-wifi#

Victim host: 10.0.0.8
Attacker host: 170.213.138.123
Block the port 4
mitigation_in
attack detected from port 4
Victim host: 10.0.0.8
Attacker host: 3.137.87.93
Block the port 4
mitigation_in
attack detected from port 4
Victim host: 10.0.0.8
Attacker host: 134.55.3.111
Block the port 4
mitigation_in
attack detected from port 4
Victim host: 10.0.0.8
Attacker host: 64.242.34.1
Block the port 4
mitigation_in
attack detected from port 4
Victim host: 10.0.0.8
Attacker host: 156.60.128.228
Block the port 4
^[[A^[[A
    
```

Figure 3-8 DDOS Attack with Detection and Mitigation



In case of DDOS Attack Detection, the controller continuously monitors the network traffic and analyzes flow statistics to identify patterns indicative of a DDoS attack. In this case, the Random Forest (RF) algorithm is used for attack detection.

In case of DDOS Attack Mitigation, When a DDoS attack is detected, the controller takes the following steps for mitigation:

- **Block Malicious Traffic:** The controller communicates with the switches and sends a flow modification message to block the traffic originating from the attackers' IP addresses (H2 and H5). The flow entry is added to the switches with a high priority and a hard timeout of 120 seconds.
- **Alert and Identify the Victim Host:** The controller logs the DDoS attack and raises an alert to indicate that a DDOS attack is in progress. By analyzing the flow statistics, the controller identifies the victim host by extracting the IP address of the attacked host (10.0.0.8) from the flow records.
- **Block Attacker's Port:** The controller identifies the port through which the attack traffic is entering the network. In this case, the port is 4 for host H2 and port 2 for host H5. The controller sends a command to the switches to block the respective ports (4 and 2) to prevent further attack traffic from entering the network.

By combining attack detection with RF algorithm and implementing effective mitigation strategies, the controller can successfully detect and mitigate the ICMP, SYN, TCP floods attacks, protecting the victim host (10.0.0.8) from the DDoS attack orchestrated by hosts H2 and H5.

### 3.5 Methodology and implementation: Man-in-the-middle Attack

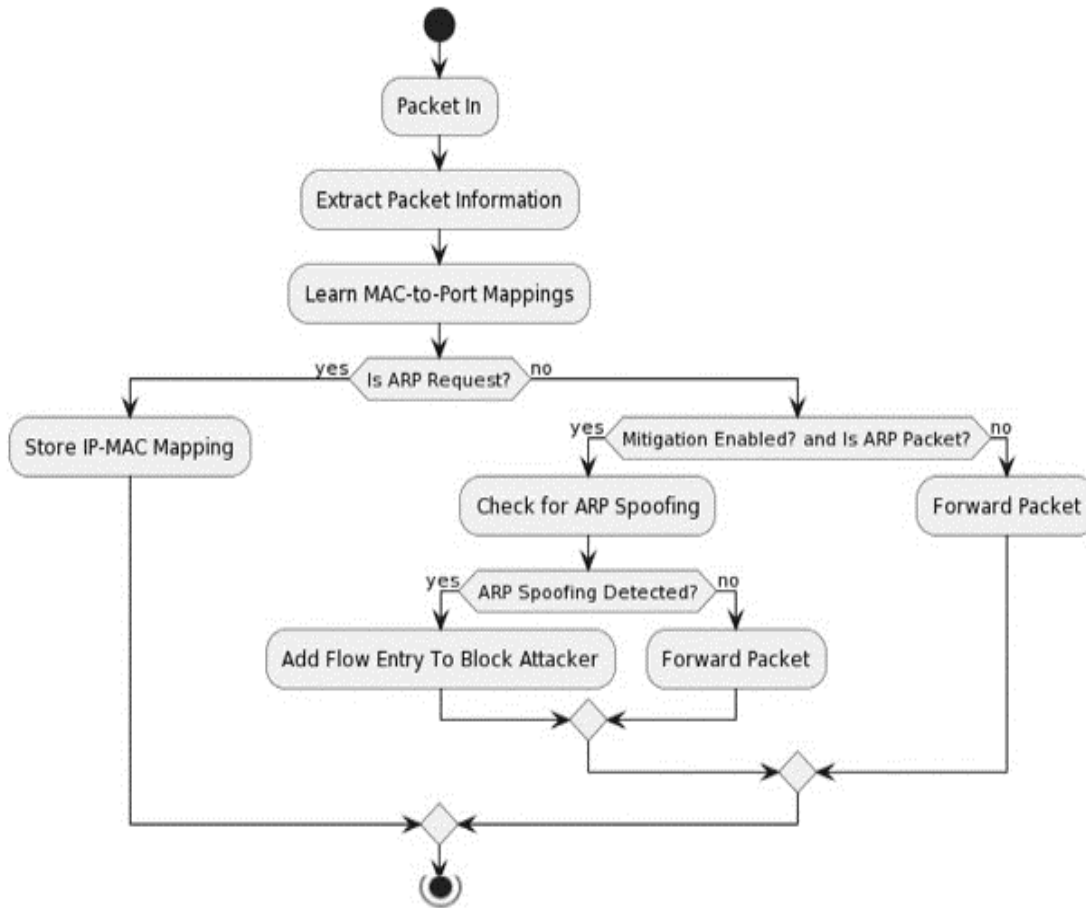
#### 3.5.1 ARP Poisoning

ARP poisoning is an attack that takes advantage of vulnerabilities in the Address Resolution Protocol (ARP) protocol used to map IP addresses to MAC addresses. In traditional networks, when one host wants to communicate with another, it sends an ARP request to determine the MAC address associated with the IP address of the destination host. All hosts on the network receive the request, and the host with the corresponding IP address responds with an ARP reply [20].

An attacker who controls a host on the network can use ARP poisoning to manipulate the ARP tables of other hosts or IoT Devices. By sending false ARP reply packets containing bogus information, the attacker can redirect traffic to their own host. In this context, the ARP table in question is not the system table of a particular host, but rather the table built inside the controller module.

To carry out an ARP poisoning attack using Ettercap, the attacker first needs to capture traffic on the network. They can then use Ettercap's built-in ARP poisoning module to create fake ARP reply packets. These packets can contain fake source and destination MAC addresses and IP addresses, allowing the attacker to redirect traffic to their own host. Alternatively, they can use Ettercap's ARP cache poisoning module to intercept and modify legitimate ARP packets.

### 3.4.2 The Proposed Algorithm for Detecting and Mitigating ARP Poisoning



**Figure 3-9 Algorithm for Detecting and Mitigating ARP Poisoning**

Upon detecting the ARP poisoning attack, the controller initiates the mitigation process. It identifies the attacker's MAC address based on the observed malicious activity.

The controller blocking the attacker's MAC address at the switch level, preventing further communication from the attacker's device. This ensures that any subsequent data packets from the victim host are not routed to the attacker.

### 3.5.3 Implementation: Second Scenario #: ARP Poisoning

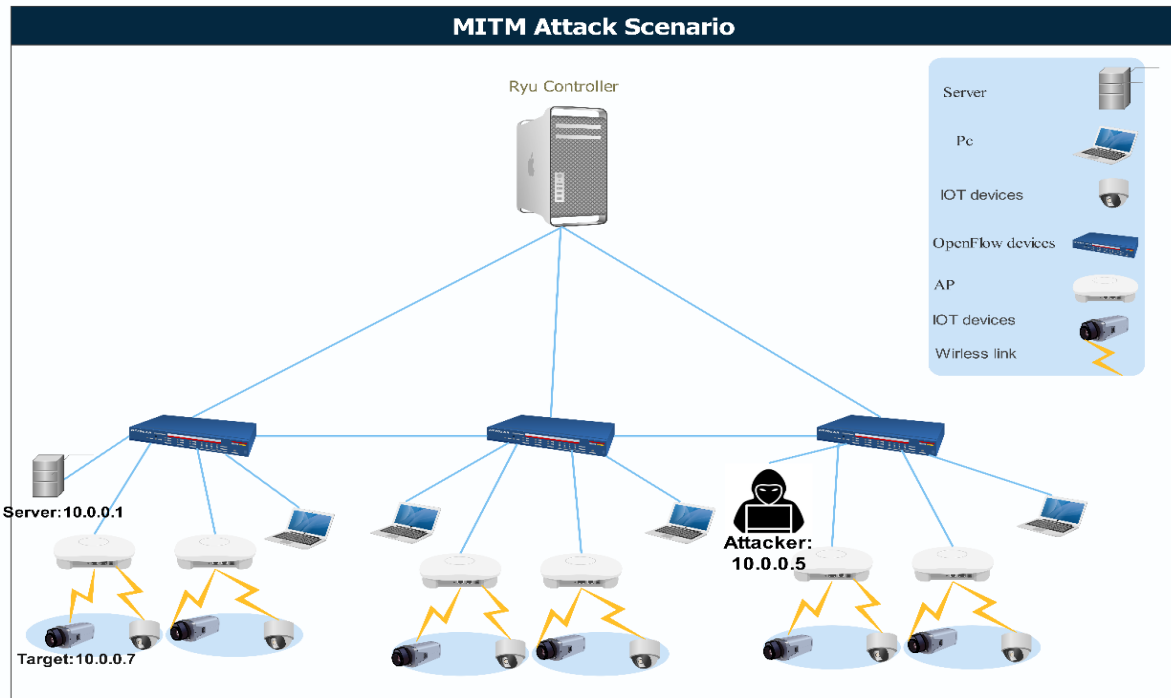


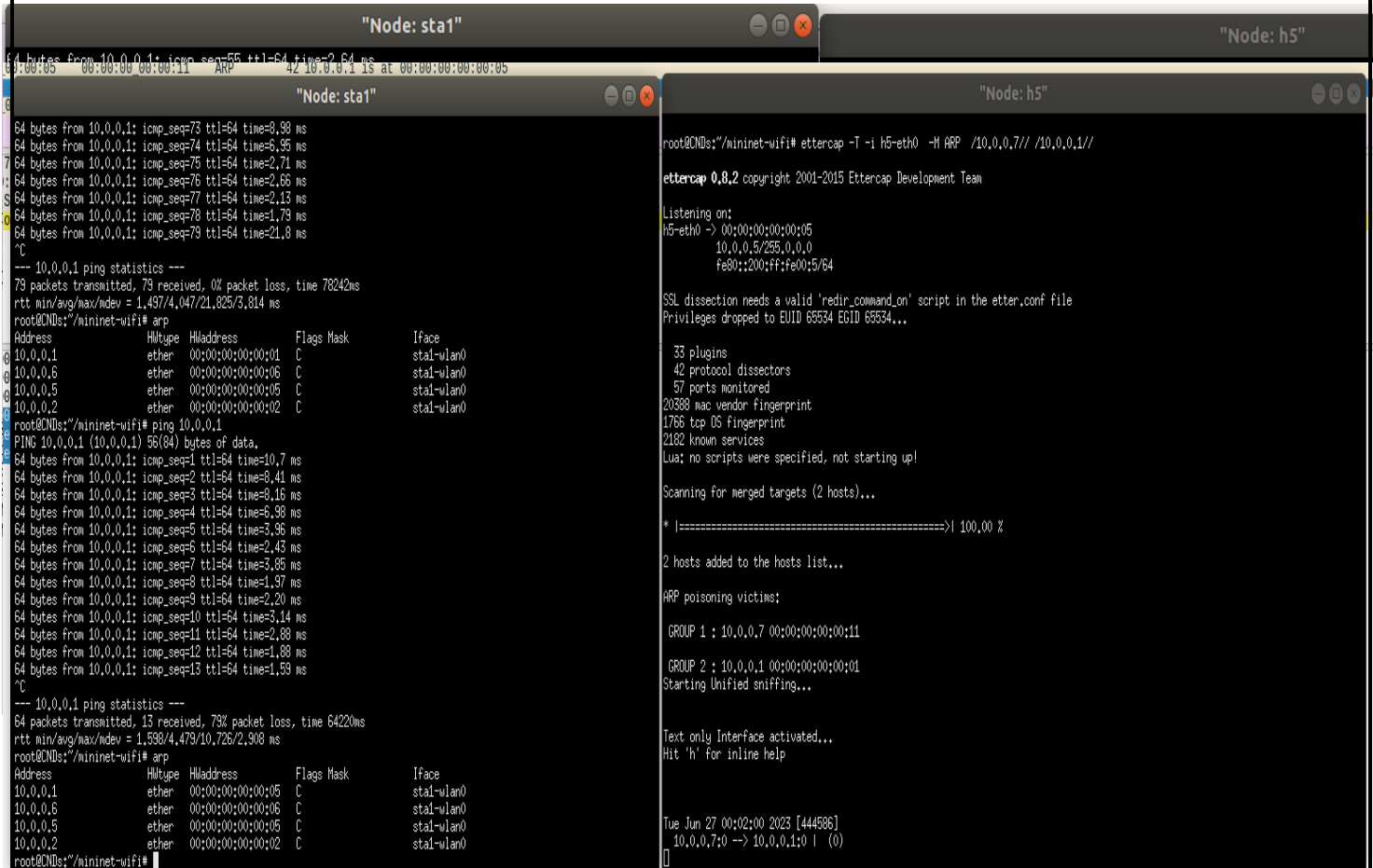
Figure 3-10 Network Topology for ARP Poisoning

In the given scenario in Figure 3-10, ARP poisoning attack is conducted by an attacker using the Ettercap tool [29]. Host No. 5 is identified as the attacker, and the victim is a specific (IOT) device with the IP address 10.0.0.7. The network also includes a server with the IP address 10.0.0.1, which is the target for data transmission from an IoT device.

The attacker's objective is to manipulate the ARP protocol by sending falsified ARP messages. This associates the attacker's MAC address with the IP address of the server (10.0.0.1). As a result, when the victim host tries to communicate with the server, the data packets are mistakenly sent to the attacker's device. Thus, the attacker gains unauthorized access to the transmitted data between the victim host and the server, posing a risk to data confidentiality, integrity, and availability.

Upon detecting the ARP poisoning attack, the controller initiates the mitigation process. It identifies the attacker's MAC address based on the observed malicious activity. The controller blocked the attacker's MAC address at the switch level, preventing further communication from the attacker's device. This ensures that any subsequent data packets from the victim host are not routed to the attacker.

Figure 3-11 Before ARP Poisoning Attack



The image shows two terminal windows. The left window, titled "Node: sta1", displays network traffic and ping statistics. The right window, titled "Node: h5", shows the configuration and execution of Ettercap on a Raspberry Pi.

```

Node: sta1
64 bytes from 10.0.0.1: icmp_seq=55 ttl=64 time=2.61 ms
42 10.0.0.1 is at 00:00:00:00:00:05

Node: sta1
64 bytes from 10.0.0.1: icmp_seq=73 ttl=64 time=8.98 ms
64 bytes from 10.0.0.1: icmp_seq=74 ttl=64 time=6.95 ms
64 bytes from 10.0.0.1: icmp_seq=75 ttl=64 time=2.71 ms
64 bytes from 10.0.0.1: icmp_seq=76 ttl=64 time=2.66 ms
64 bytes from 10.0.0.1: icmp_seq=77 ttl=64 time=2.13 ms
64 bytes from 10.0.0.1: icmp_seq=78 ttl=64 time=1.79 ms
64 bytes from 10.0.0.1: icmp_seq=79 ttl=64 time=21.8 ms
^C
-- 10.0.0.1 ping statistics --
79 packets transmitted, 79 received, 0% packet loss, time 7824ms
rtt min/avg/max/ndev = 1.497/4.047/21.825/3.814 ms
root@CNDS:/mininet-wifi# arp
Address      Hwtype  Hwaddress      Flags Mask      Iface
10.0.0.1    ether   00:00:00:00:00:01  C           10.0.0.1
10.0.0.6    ether   00:00:00:00:00:06  C           stal-wlan0
10.0.0.5    ether   00:00:00:00:00:05  C           stal-wlan0
10.0.0.2    ether   00:00:00:00:00:02  C           stal-wlan0
root@CNDS:/mininet-wifi# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=10.7 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=8.41 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=8.16 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=6.98 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=3.96 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=2.43 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=3.85 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=1.97 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=2.20 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=3.14 ms
64 bytes from 10.0.0.1: icmp_seq=11 ttl=64 time=2.88 ms
64 bytes from 10.0.0.1: icmp_seq=12 ttl=64 time=1.88 ms
64 bytes from 10.0.0.1: icmp_seq=13 ttl=64 time=1.93 ms
^C
-- 10.0.0.1 ping statistics --
64 packets transmitted, 13 received, 79% packet loss, time 64220ms
rtt min/avg/max/ndev = 1.598/4.479/10.726/2.908 ms
root@CNDS:/mininet-wifi# arp
Address      Hwtype  Hwaddress      Flags Mask      Iface
10.0.0.1    ether   00:00:00:00:00:05  C           10.0.0.1
10.0.0.6    ether   00:00:00:00:00:06  C           stal-wlan0
10.0.0.5    ether   00:00:00:00:00:05  C           stal-wlan0
10.0.0.2    ether   00:00:00:00:00:02  C           stal-wlan0
root@CNDS:/mininet-wifi#

Node: h5
root@CNDS:/mininet-wifi# ettercap -T -i h5-eth0 -H ARP /10.0.0.7//10.0.0.1//
ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team

Listening on:
h5-eth0 -> 00:00:00:00:00:05
          10.0.0.5/255.0.0.0
          fe80::200:ff:fe00:5/64

SSL dissection needs a valid 'redir_command_on' script in the etter_conf file
Privileges dropped to EUID 65534 EUID 65534...

33 plugins
42 protocol dissectors
57 ports monitored
20388 mac vendor fingerprint
1766 top OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Scanning for merged targets (2 hosts)...

* |=====| 100,00 %

2 hosts added to the hosts list...

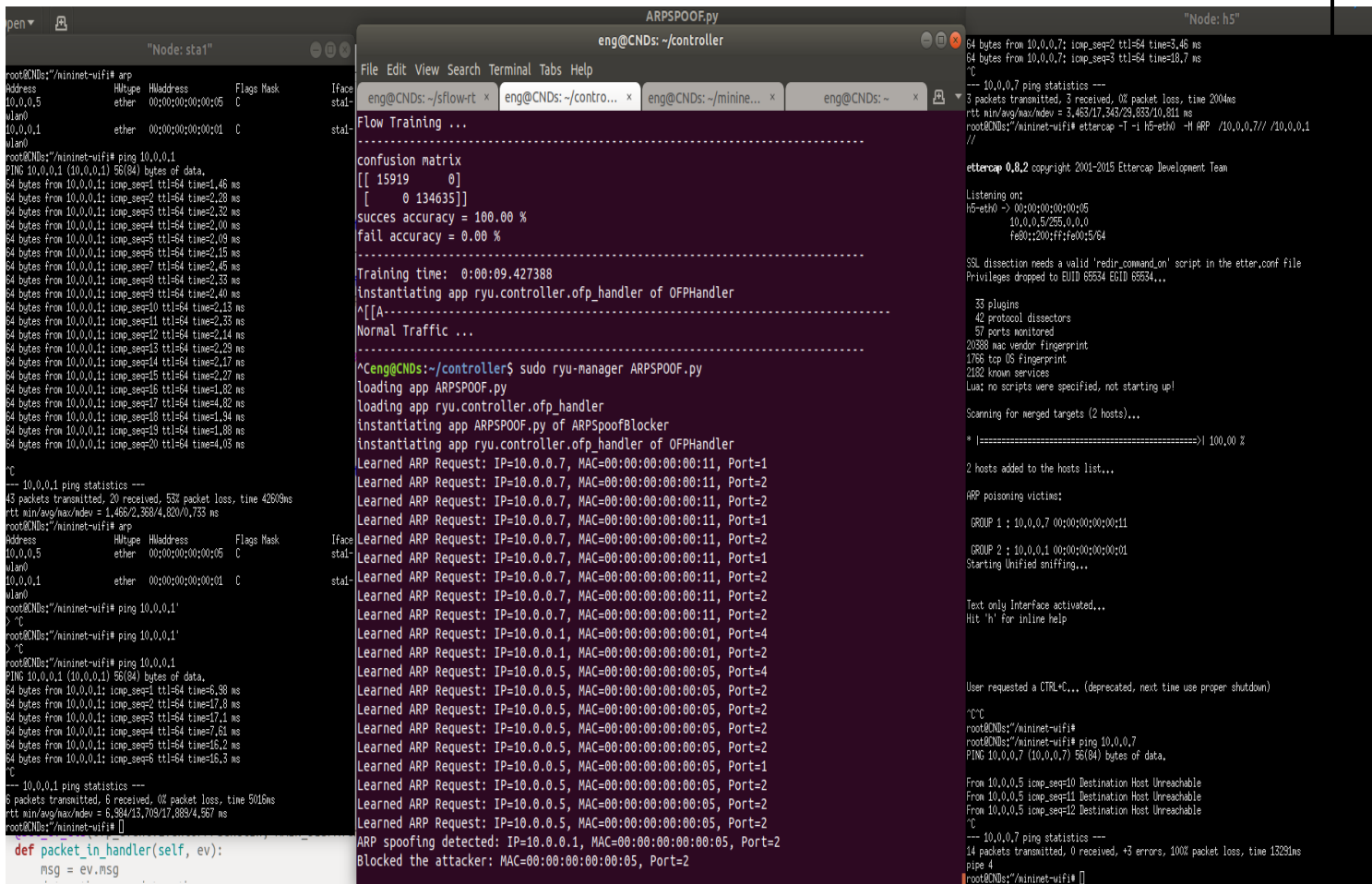
ARP poisoning victims:

GROUP 1 : 10.0.0.7 00:00:00:00:00:11
GROUP 2 : 10.0.0.1 00:00:00:00:00:01
Starting Unified sniffing...

Text only Interface activated...
Hit 'h' for inline help

Tue Jun 27 00:02:00 2023 [444586]
10.0.0.7:0 -> 10.0.0.1:0 | (0)
    
```

Figure 3-12 during ARP Poisoning Attack



```

root@CND5:~/mininet-wifi# arp
Address      HwType HwAddress      Flags Mask      Iface
10.0.0.5     ether  00:00:00:00:00:06 C              C
wlan0
10.0.0.1     ether  00:00:00:00:00:01 C              C
wlan0

root@CND5:~/mininet-wifi# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=1.46 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=2.38 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=2.32 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=2.00 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=2.09 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=2.15 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=2.45 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=2.35 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=2.40 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=2.13 ms
64 bytes from 10.0.0.1: icmp_seq=11 ttl=64 time=2.33 ms
64 bytes from 10.0.0.1: icmp_seq=12 ttl=64 time=2.14 ms
64 bytes from 10.0.0.1: icmp_seq=13 ttl=64 time=2.29 ms
64 bytes from 10.0.0.1: icmp_seq=14 ttl=64 time=2.17 ms
64 bytes from 10.0.0.1: icmp_seq=15 ttl=64 time=2.27 ms
64 bytes from 10.0.0.1: icmp_seq=16 ttl=64 time=1.82 ms
64 bytes from 10.0.0.1: icmp_seq=17 ttl=64 time=1.82 ms
64 bytes from 10.0.0.1: icmp_seq=18 ttl=64 time=1.94 ms
64 bytes from 10.0.0.1: icmp_seq=19 ttl=64 time=1.88 ms
64 bytes from 10.0.0.1: icmp_seq=20 ttl=64 time=1.03 ms
^C
--- 10.0.0.1 ping statistics ---
43 packets transmitted, 20 received, 53% packet loss, time 4263ms
rtt min/avg/max/mdev = 1.466/2.388/4.820/0.733 ms

root@CND5:~/mininet-wifi# arp
Address      HwType HwAddress      Flags Mask      Iface
10.0.0.5     ether  00:00:00:00:00:06 C              C
wlan0
10.0.0.1     ether  00:00:00:00:00:01 C              C
wlan0

root@CND5:~/mininet-wifi# ping 10.0.0.1
> ^C
root@CND5:~/mininet-wifi# ping 10.0.0.1
> ^C
root@CND5:~/mininet-wifi# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=6.98 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=17.8 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=17.1 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=17.61 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=16.2 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=16.3 ms
^C
--- 10.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 501ms
rtt min/avg/max/mdev = 6.384/13.793/17.893/4.567 ms

root@CND5:~/mininet-wifi# [

eng@CND5:~/controller
File Edit View Search Terminal Tabs Help
eng@CND5:~/sflow-rt x eng@CND5:~/contro... x eng@CND5:~/minine... x eng@CND5:~/... x
Flow Training ...
-----
confusion matrix
[[ 15919    0]
 [    0 134635]]
succes accuracy = 100.00 %
fail accuracy = 0.00 %
-----
Training time: 0:00:09.427388
instantiating app ryu.controller.ofp_handler of OFPHandler
^[[A-----
Normal Traffic ...
-----
^Ceng@CND5:~/controller$ sudo ryu-manager ARPSPOOF.py
loading app ARPSPOOF.py
loading app ryu.controller.ofp_handler
instantiating app ARPSPOOF.py of ARPSpoofBlocker
instantiating app ryu.controller.ofp_handler of OFPHandler
Learned ARP Request: IP=10.0.0.7, MAC=00:00:00:00:00:11, Port=1
Learned ARP Request: IP=10.0.0.7, MAC=00:00:00:00:00:11, Port=2
Learned ARP Request: IP=10.0.0.7, MAC=00:00:00:00:00:11, Port=1
Learned ARP Request: IP=10.0.0.7, MAC=00:00:00:00:00:11, Port=2
Learned ARP Request: IP=10.0.0.7, MAC=00:00:00:00:00:11, Port=1
Learned ARP Request: IP=10.0.0.7, MAC=00:00:00:00:00:11, Port=2
Learned ARP Request: IP=10.0.0.7, MAC=00:00:00:00:00:11, Port=2
Learned ARP Request: IP=10.0.0.7, MAC=00:00:00:00:00:11, Port=2
Learned ARP Request: IP=10.0.0.7, MAC=00:00:00:00:00:11, Port=2
Learned ARP Request: IP=10.0.0.5, MAC=00:00:00:00:00:05, Port=2
Learned ARP Request: IP=10.0.0.5, MAC=00:00:00:00:00:05, Port=2
Learned ARP Request: IP=10.0.0.5, MAC=00:00:00:00:00:05, Port=2
Learned ARP Request: IP=10.0.0.5, MAC=00:00:00:00:00:05, Port=2
Learned ARP Request: IP=10.0.0.5, MAC=00:00:00:00:00:05, Port=2
Learned ARP Request: IP=10.0.0.5, MAC=00:00:00:00:00:05, Port=2
Learned ARP Request: IP=10.0.0.5, MAC=00:00:00:00:00:05, Port=2
Learned ARP Request: IP=10.0.0.5, MAC=00:00:00:00:00:05, Port=2
ARP spoofing detected: IP=10.0.0.1, MAC=00:00:00:00:00:05, Port=2
Blocked the attacker: MAC=00:00:00:00:00:05, Port=2

ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team

Listening on:
h5-eth0 -> 00:00:00:00:00:06
10.0.0.5/255.0.0.0
fe80::220::ff:fe00:5/64

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EGID 65534,...

33 plugins
42 protocol dissectors
57 ports monitored
20388 mac vendor fingerprint
1768 top OS fingerprint
2182 known services
Load no scripts were specified, not starting up!

Scanning for merged targets (2 hosts) ...

* |=====| 100,00 %

2 hosts added to the hosts list...

ARP poisoning victim:

GROUP 1 : 10.0.0.7 00:00:00:00:00:11

GROUP 2 : 10.0.0.1 00:00:00:00:00:01

Starting Unified sniffing...

Text only interface activated...
hit 'h' for inline help

User requested a CTRL+C... (deprecated, next time use proper shutdown)

^C^C
root@CND5:~/mininet-wifi#
root@CND5:~/mininet-wifi# ping 10.0.0.7
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data:
From 10.0.0.5 icmp_seq=10 Destination Host Unreachable
From 10.0.0.5 icmp_seq=11 Destination Host Unreachable
From 10.0.0.5 icmp_seq=12 Destination Host Unreachable
^C
--- 10.0.0.7 ping statistics ---
14 packets transmitted, 0 received, 100% packet loss, time 1329ms
pipe 4

root@CND5:~/mininet-wifi# [
    
```

**Figure 3-13 ARP Poisoning Attack with Mitigation**

In Fig. 3-13, As a result of the mitigation and blocking mechanism, the ARP table in sta1 (the victim host) is not updated with the falsified IP-MAC mapping. This ensures that sta1 does not mistakenly send data packets to the attacker's device. Thus, protecting the integrity and confidentiality of the communication between the victim host and the server.

### 3.6 Simulation Environment

The Simulation Environment involved setting up the network infrastructure, installing the required software and tools, configuring the network devices, and programming the SDN controller. The implementation was done using the Ubuntu 18.04 OS VM, SDN Ryu [22] controller use Python language, Mininet-WiFi [28] to simulate a public network containing IoT devices, regular computers, and access point devices needed to build the target network, and the OpenFlow protocol to implement SDN.

### 3.6.1 Development and Simulation Platform Tools

In this work, the algorithm is implemented in the virtual environment that is created by the VirtualBox. Ubuntu 18.04 is installed for creating the OS environment for the simulation.

The following tools and technologies were used to implement the methodology:

1. **OpenFlow:** It is a protocol for software-defined networking (SDN) that separates control and data planes, enabling centralized control of network devices.
2. **Mininet-WiFi** [28]: it is an open-source network emulator that combines Mininet, a network emulator, with the wireless simulation capabilities of the OpenWrt software. It allows researchers and developers to create virtual network topologies with support for WiFi and SDN (Software-Defined Networking) elements. Mininet-WiFi enables the evaluation and testing of various network protocols and applications in a controlled environment.
3. **Ryu Controller** [22]: Ryu is an open-source framework for developing SDN applications and controllers using the OpenFlow protocol, provides a set of APIs and libraries for easy integration with OpenFlow-enabled devices. It is written in Python, making it accessible to a wide range of developers and enabling rapid prototyping of SDN applications.
4. **Hping3:** it is a packet generator tool that allows users to generate TCP/IP traffic within a network. It is commonly employed in network security testing, as it enables the creation of both normal and attack traffic scripts. These scripts can be utilized to automatically generate traffic, aiding in the evaluation of network security measures.
5. **Iperf** [30]: it is an open-source tool for measuring network bandwidth and performance by generating TCP and UDP traffic between endpoints.
6. **Ettercap** [29]: it is a comprehensive network sniffing and man-in-the-middle (MITM) attack tool widely used for security assessments and penetration testing. It provides features for monitoring and analyzing network traffic, performing various attacks like ARP spoofing, and capturing sensitive information transmitted over the network. Ettercap supports multiple protocols and is known for its versatility and ease of use.
7. **SFlow-rt** [31]: it is a software package used for real-time network monitoring and analytics. It collects SFlow-rt data from network devices and provides insights into network traffic, performance, and security. SFlow-rt offers features such as flow sampling, traffic visualization, anomaly detection, and integration with other monitoring tools. It is particularly useful for managing large-scale networks.
8. **NMON** [32]: it is a performance monitoring tool for Unix-like systems, providing system-level monitoring and reporting of CPU, memory, disk I/O, network activity, and other metrics.

## CHAPTER 4 RESULTS AND DISCUSSION

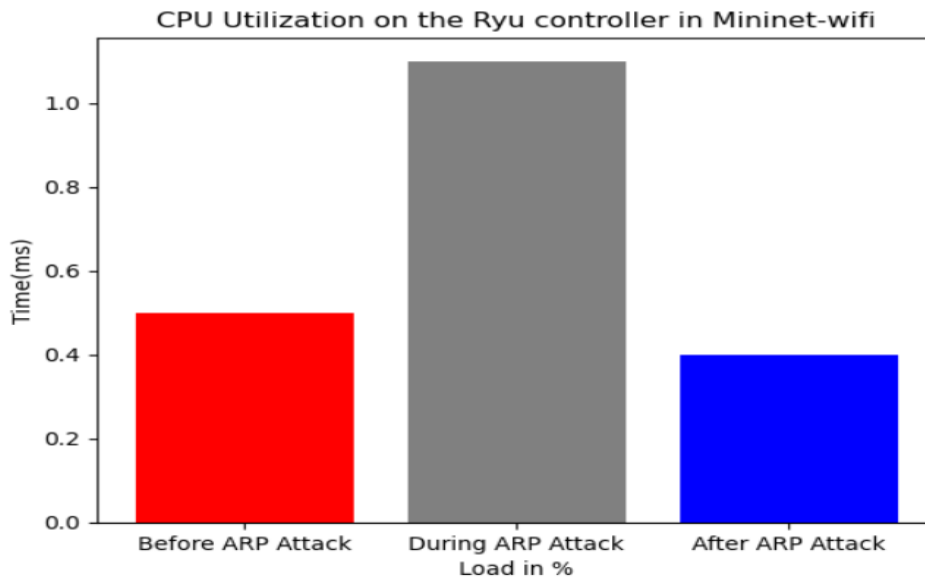
### 4.1 Overview

In This chapter discusses the results that we have get it from the implementation stage SDN Network using Mininet-WiFi for implement the algorithms for detecting and mitigating MiTM, DDOS Attacks in Ryu Controller with its results to make it easier to be understood and discussed.

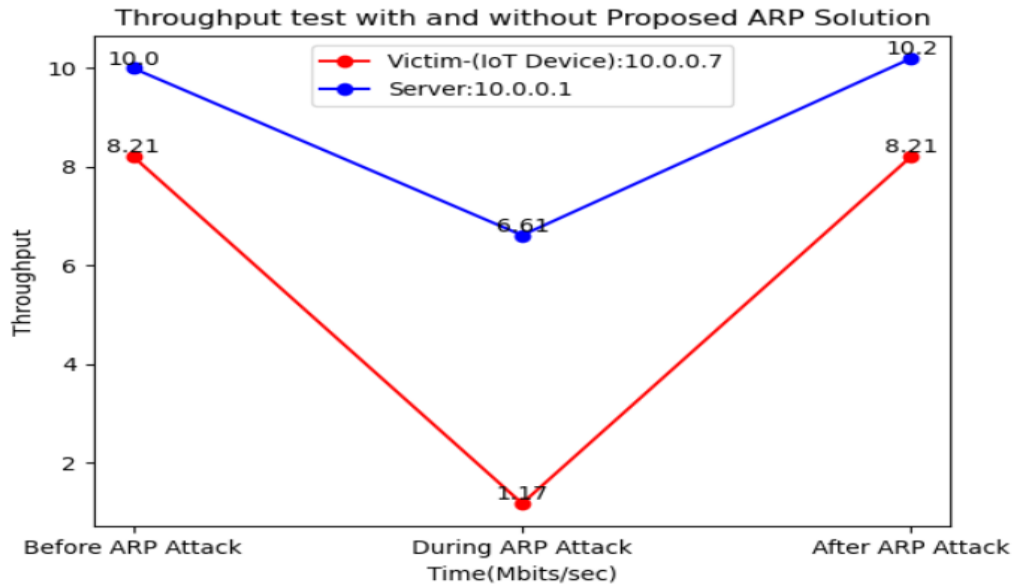
## 4.2 Results of Scenario #: ARP Poisoning

In Fig.4-1. The results of the experiments are highly promising, as they demonstrate minimal or negligible impact on the controller's performance. the execution of the Proposed ARP algorithm did not significantly burden the controller or cause any noticeable slowdown in its operations.

**Figure 4-1 CPU Utilization on Ryu Controller**



To evaluate the performance of the network under different conditions, a throughput test was conducted on the topology depicted by Iperf [30] and NMON [32] tools to collect the results in Fig.4-2. The objective of the test was to measure the TCP connection throughput at different stages: before, during, and after the attack. The throughput refers to the amount of data that can be transmitted over a network connection within a given time. Fig.4-2 presents a graphical representation of the test results, illustrating the varying TCP connection throughput values at different stages of the experiment. By comparing the throughput values before, during, and after the attack, insights can be gained into the network's ability to handle data transmission under normal and attack conditions.



*Figure 4-2 Throughput test between Server and Victim*

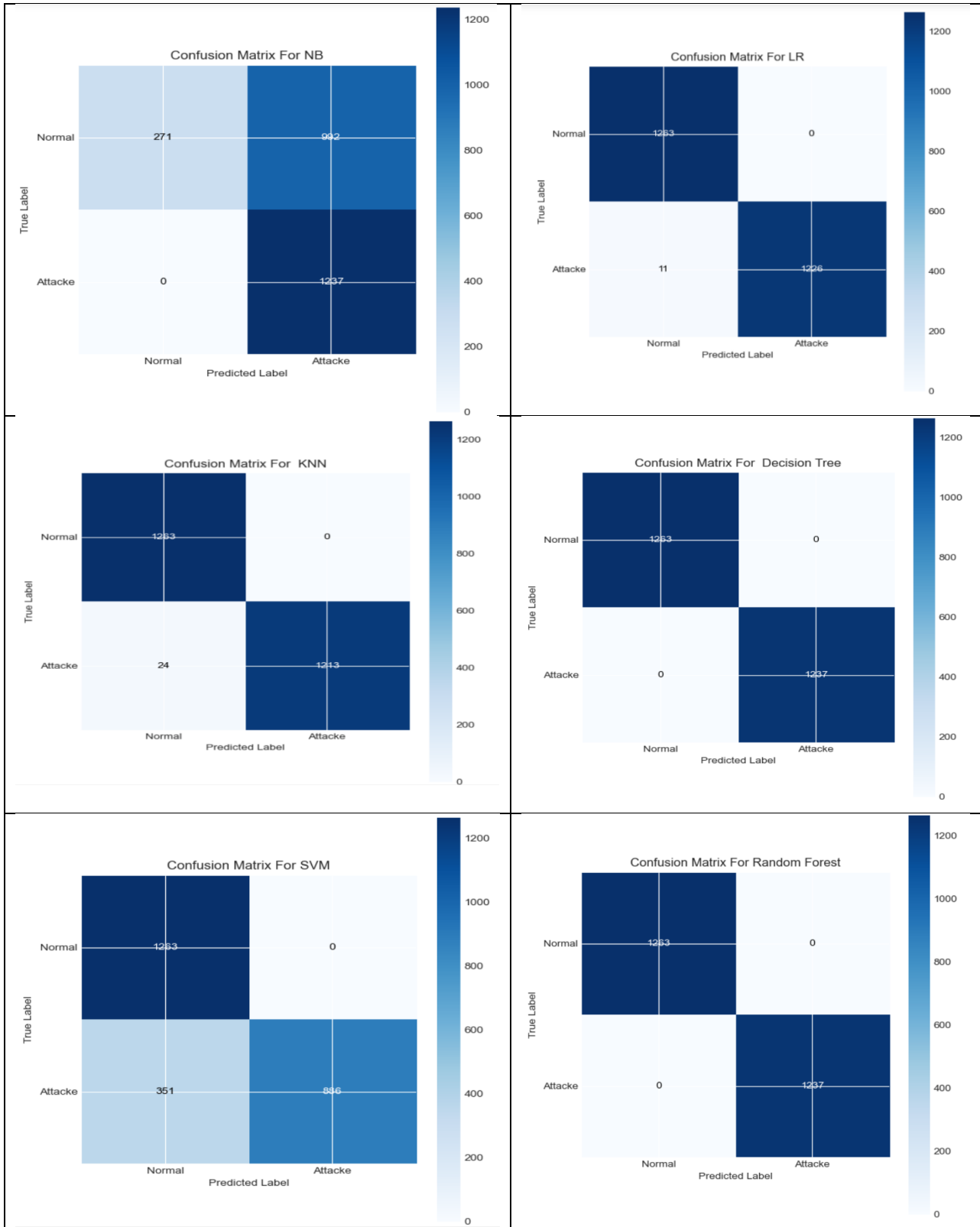
### 4.3 Results of Machine Learning Algorithms

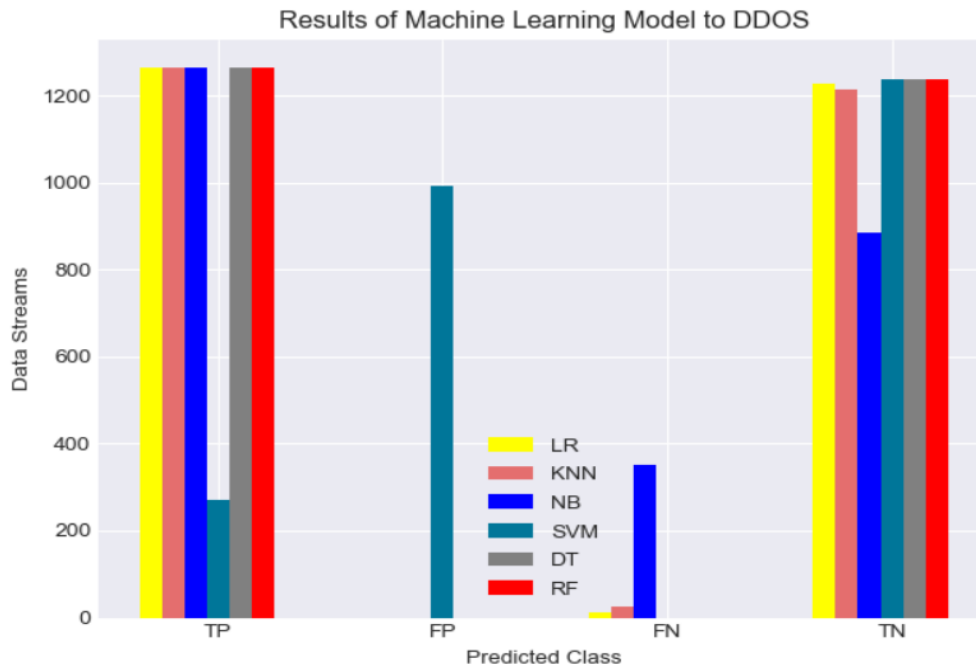
*Table 4-1 Results of Machine Learning Algorithms*

ML Model	Accuracy	TP	FP	FN	TN	Time in sec
1) LR	99.56%	1263	0	11	1226	0.1620s
2) KNN	99.04%	1263	0	24	1213	0.9657s
3) SVM	85.96%	1263	0	351	886	5.1944s
4) NB	60.32%	271	0	0	1237	0.0495s
5) DT	100%	1263	0	0	1237	0.509s
6) RF	100%	1263	0	0	1237	0.4319s

*Table 4-2 Results of Confusion Matrix for ML Algorithms*

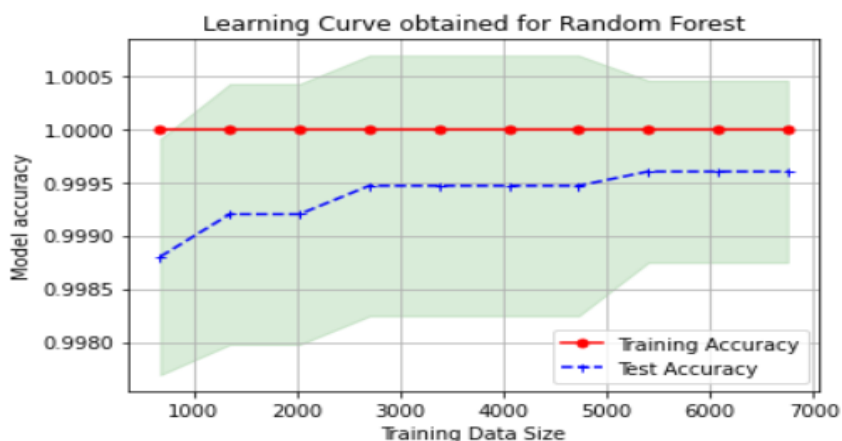






**Figure 4-3 Results of ML Models to DDOS**

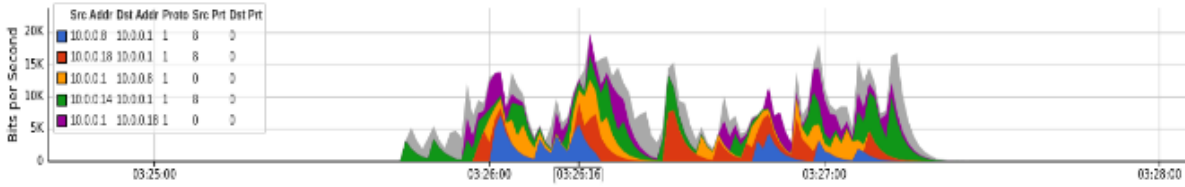
In Tables 4-1, 4-2, and Fig. 4-3. the results of the ML Model are highly promising, as they demonstrate minimal or negligible impact on the controller's performance. In other words, the execution of the machine learning algorithms did not significantly burden the controller or cause any noticeable slowdown in its operations, thus, the best model of Machine Learning is Random Forest.



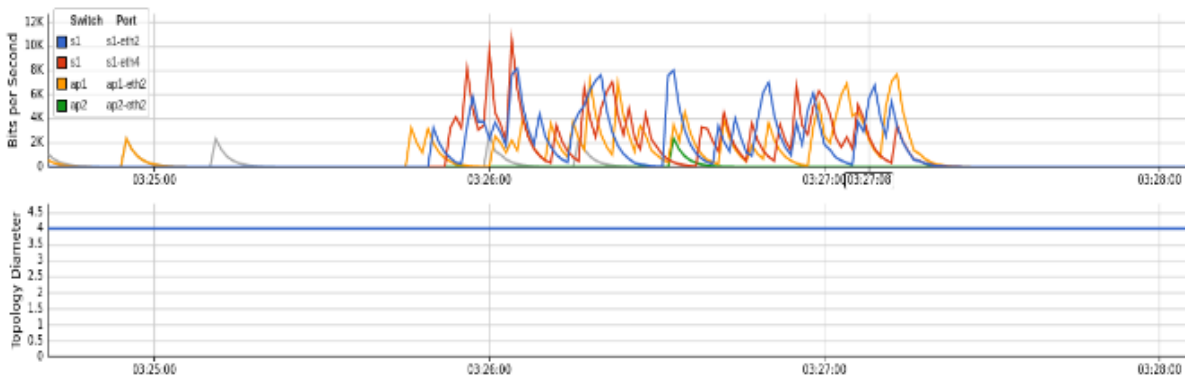
**Figure 4-4 Result of RF Machine Learning Model**

### 4.3 Results of Second Scenario: DDOS

▼ Traffic

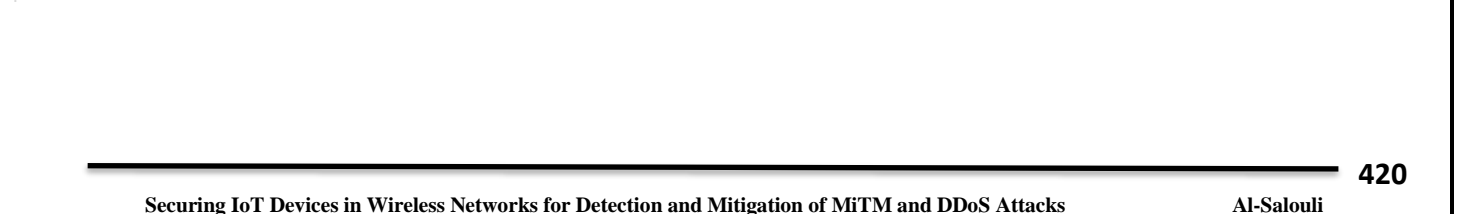
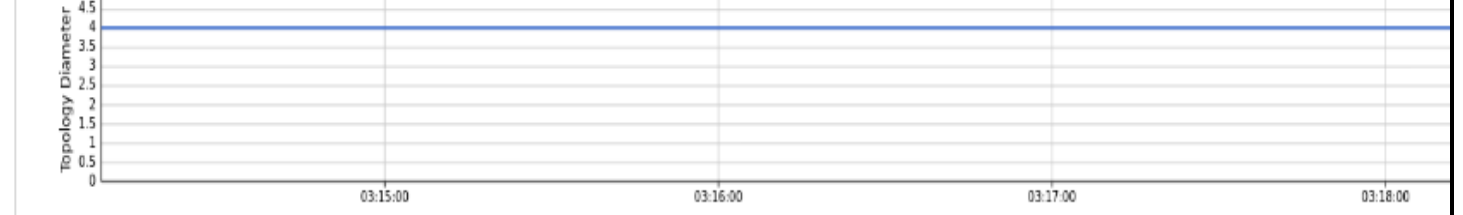
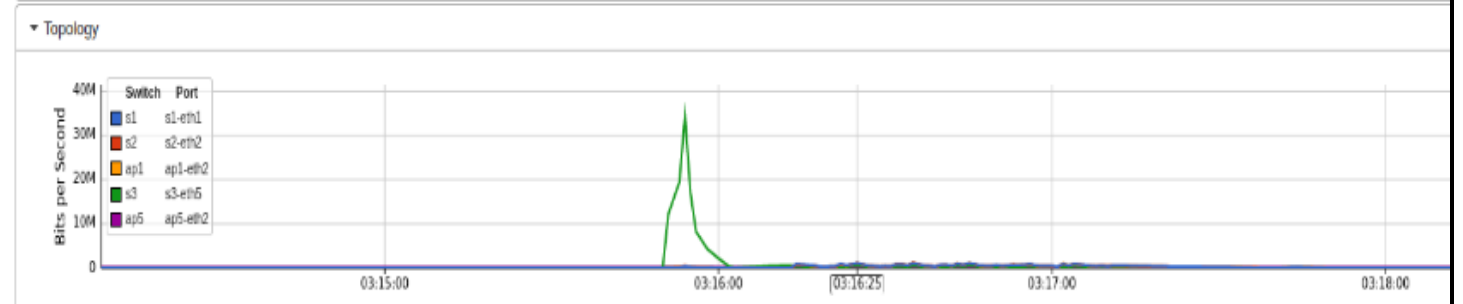
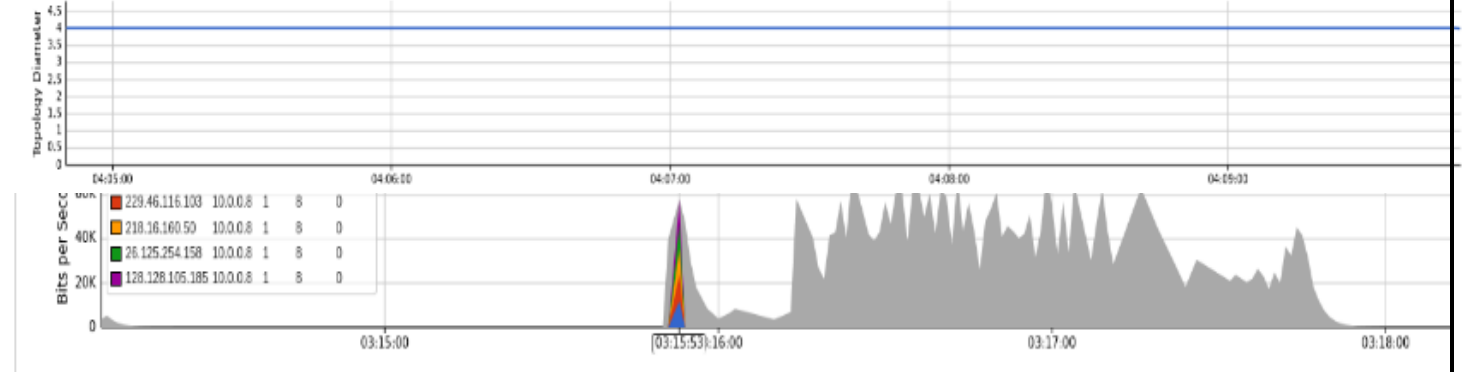
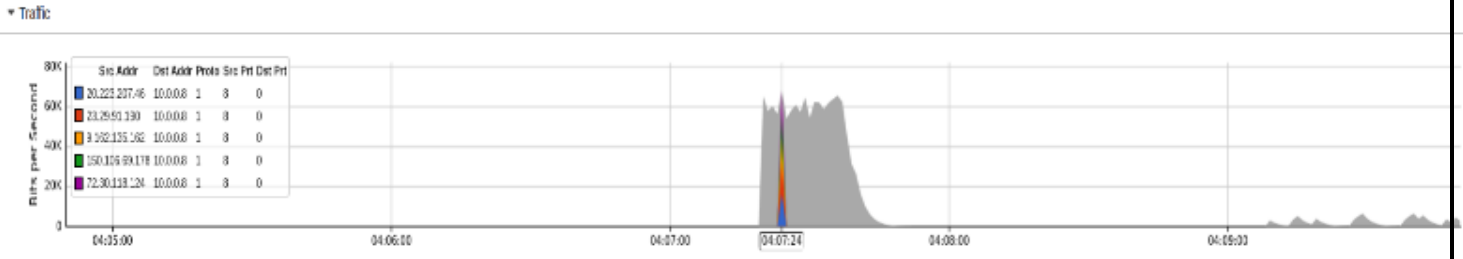


▼ Topology

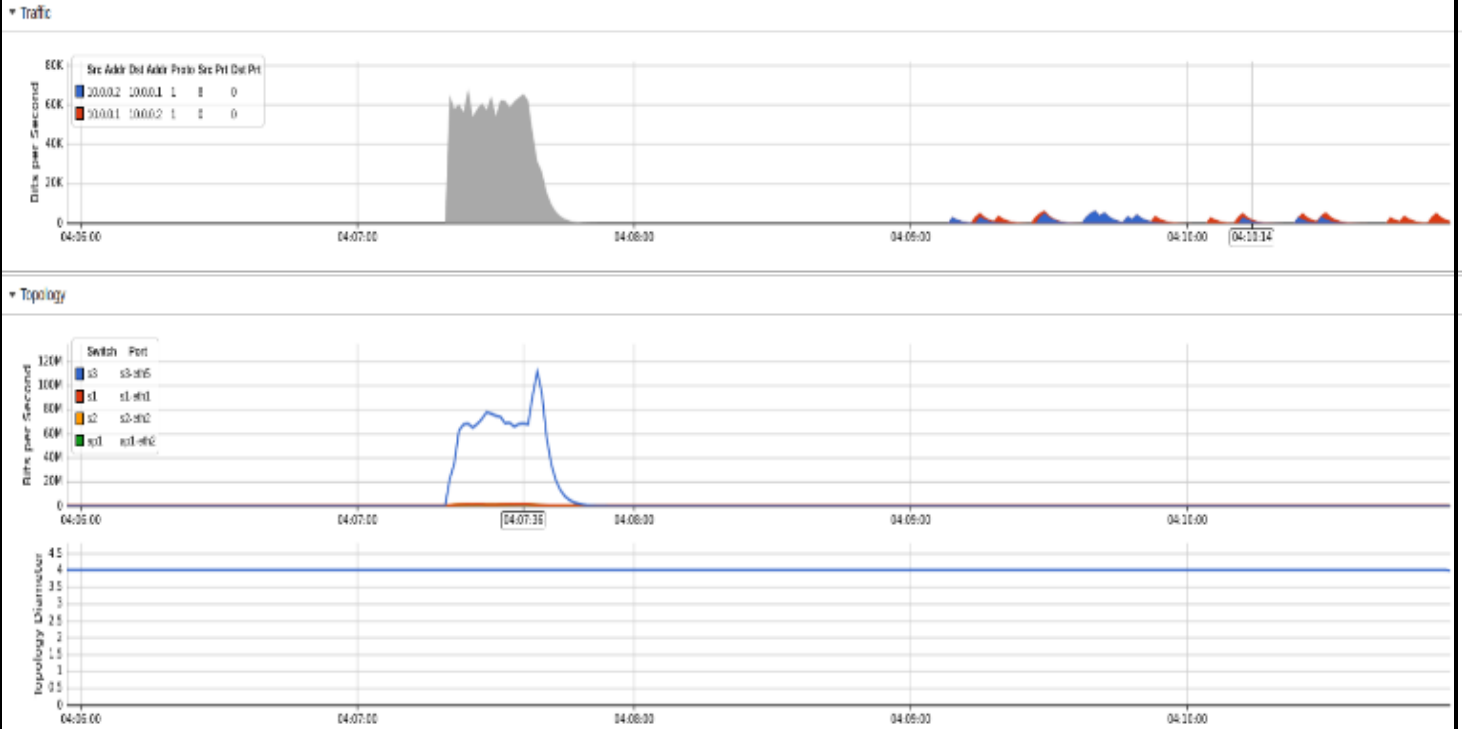


**Figure 4-5 Normal Traffic of Network**

**Figure 4-6 DDOS Traffic**



**Figure 4-7 DDOS Traffic with Mitigation**



**Figure 4-8 Different between Normal Traffic and DDOS Traffic with Mitigation**

In Figures [4-5 to 4-8]. The results of the experiments in Traffic, Topology, Mitigation of DDOS Attack by sFlow-rt [31] tool are highly promising, as they demonstrate minimal or negligible impact on the controller's performance and Mininet-WiFi [28] IoT devices. the execution of the Proposed Random Forest algorithm in detection and mitigation DDOS did not significantly burden the controller or cause any noticeable large slowdown in its operations.

#### 4.4 Evaluation of DDOS

In this section, we present a comparison of the final results obtained from our implementation, as shown in Table 4-3, with a study conducted by Naman et al [14] in 2023. We chose this study for comparison because it similar several similarities with our work in DDOS, including the use of similar machine learning methods, feature selection techniques, and a dataset. However, as a contribution to the existing literature, we have introduced novel features to enhance the effectiveness of the approach.

**Table 4-3 Comparison of the performance of two exercises against the evaluation criteria.**

ML Algorithms	Our study	Naman et al [14]
	Accuracy	Accuracy
1) LR	99.56%	66.02%

2) KNN	99.04%	100%
3) SVM	85.96%	83.58%
4) RF	100%	100%
5) DT	100%	100%
6) NB	60.32%	71.41%

## CHAPTER 5 CONCLUSION AND FUTURE WORK

### 5.1 Conclusion

The research focuses on techniques to detect and mitigate MITM, DDoS attacks in Internet of Things (IoT) devices within Software-Defined Networking (SDN) networks. The proposed approach involves in DDOS using ML algorithms to analyze network traffic and identify patterns associated with DDoS attacks. and in ARP Poisoning When an Address Resolution Protocol (ARP) request is encountered, the system stores the IP-MAC mapping and checks for ARP spoofing if mitigation is enabled. If an ARP spoofing attack is detected, the system adds a flow entry to block the attacker; otherwise, the packet is forwarded. When mitigation is disabled, packets are directly forwarded. Regular removal of expired flow entries ensures optimal network performance and security by eliminating outdated or irrelevant entries. The research demonstrates that ML algorithms effectively detect and mitigate DDoS attacks in SDN-based IoT networks. thereby enhancing the security and resilience of IoT devices.

### 5.2 Future Work

Future research could focus on enhancing the ML-based detection and mitigation system by exploring advanced ML algorithms or incorporating real-time feedback mechanisms. Additionally, investigating the scalability and performance aspects of the system would be valuable to ensure its effectiveness in large-scale IoT-SDN deployments.

### References

- Gan, G., Lu, Z., & Jiang, J. (2011). Internet of things security analysis. *IEEE*, 1–4.
- Opennetworking.org. (n.d.). Software-defined networking: The new norm for networks. Retrieved September 27, 2023, from <https://www.opennetworking.org/>
- Suh, M., Park, S., Lee, B., & Yang, S. (2014). Building firewall over the software-defined network controller. *IEEE*, 744–748.
- Chippalkatti, O., & Nimbhorkar, S. U. (2017). An approach for detection of attacks in software-defined networks. In *2017 International Conference on Innovations in Information Technology*.

Sood, K., Yu, S., & Xiang, Y. (2015). Software-defined wireless networking: Opportunities and challenges for Internet-of-Things: A review. *IEEE Internet of Things Journal*, 2(4), 453–463.

Kodi.tv. (n.d.). Open source home theatre software. Retrieved September 28, 2023, from <http://kodi.tv/>

AbdelSalam, A. M., El-Sisi, A. B., & Reddy, V. (2015). Mitigating ARP spoofing attacks in software-defined networks. In *2015 25th International Conference on Computer Theory and Applications (ICCTA)* (pp. 126–131). Alexandria, Egypt.

Hayajneh, A. A., Bhuiyan, M. Z. A., & McAndrew, I. (2020). Improving Internet of Things (IoT) security with software-defined networking (SDN). *IEEE*, 103, 2–6.

Bhunia, S. S., & Gurusamy, M. (2017). Dynamic attack detection and mitigation in IoT using SDN. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)* (pp. 1–6). Melbourne, VIC, Australia.

Karmakar, K. K., Varadharajan, V., Nepal, S., & Tupakula, U. (2020). SDN enabled secure IoT architecture. *IEEE*, 5, 2–5.

Saritakumar, N., Anusuya, V. K., & Balasaraswathi, B. (2021). Detection and mitigation of MITM attack in software-defined networks. In *International Conference on Combinatorial and Optimization (ICCAP)*.

Mohsin, M. A., & Hamad, A. H. (2022). Performance evaluation of SDN DDoS attack detection and mitigation based on random forest and K-nearest neighbors machine learning algorithms. *Revue d'Intelligence Artificielle*, 36(2), 233–240.

Ndayizigiye, J. P. (2023). Utilizing machine learning algorithms for early detection of DDoS attacks in software-defined networking. *Journal of Emerging Technologies and Innovative Research (JETIR)*, 10(5), 1–13.

Sonthalia, N., Reddy, E. V. A., Pagaria, H., & Jayasri, G. V. (2023). Using machine learning in software-defined networks to recognize and avoid DDoS attacks. *Ijrasnet Journal for Research in Applied Science and Engineering Technology*, 11(III), 1–9.

Ranger, S. (2020, February 3). What is the IoT? Everything you need to know about the Internet of Things right now. *ZDNet*. Retrieved November 10, 2022, from <https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/>

Reynolds, I. J. (2020, July 23). Internet of things (IoT) architecture. *Zibtek*. Retrieved November 12, 2022, from <https://www.zibtek.com/blog/iot-architecture/>

Bizanis, N., & Kuipers, F. (2016). SDN and virtualization solutions for the Internet of Things: A survey. *IEEE*, 4, 5591–5606.

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *IEEE*, 103, 14–76.

Open Networking Foundation. (n.d.). Open Networking Foundation (ONF). Retrieved November 14, 2022, from <https://www.opennetworking.org/>

Open Networking Foundation. (n.d.). *Open Networking Foundation (ONF)*. <https://www.opennetworking.org/>

Imperva. (n.d.). *ARP spoofing*. <https://www.imperva.com/learn/application-security/arp-spoofing/>

Cloudflare. (n.d.). *What is DDoS attack?* <https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/>

OSRG. (2017). *Ryu SDN controller*. <https://osrg.github.io/ryu/>

Linkletter, B. (2015, April 10). *Using the Ryu SDN controller*. <https://www.brianlinkletter.com/2015/04/using-the-pox-sdn-controller/>

Mohammed, M., Khan, M. B., & Bashier, E. (2016). In *Machine learning: Algorithms and applications*.

Sathya, R., & Abraham, A. (2013). Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 2(2), 34-38.

Géron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Inc.

Varghese, D. (2018, December 6). *Comparative study on classic machine learning algorithms*. <https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222>

Mininet-wifi. (n.d.). *Mininet-wifi tool*. <http://mininet-wifi.org/>

Valleri, A., & Ornaghi, M. (n.d.). *Ettercap*. <http://ettercap.github.io/ettercap>

Stone, J. (n.d.). *iperf: The TCP/UDP bandwidth measurement tool*. <https://iperf.fr/>

InMon Corp. (n.d.). *sFlow-RT: Real-time traffic analysis software*. <https://inmon.com/products/sflow-rt/>

nmon.sourceforge.net. (n.d.). *IBM Power Systems Performance Monitor (nmon)*. <http://nmon.sourceforge.net/pmwiki.php>

Kodi. (n.d.). *Open source home theatre software*. <http://kodi.tv/>

Open Networking Foundation. (n.d.). *Software-defined networking: The new norm for networks*. <https://www.opennetworking.org/>