

A FRAMEWORK FOR DETECTING AND MITIGATING DDOS ATTACKS IN SOFTWARE-DEFINED IOT NETWORKS

by

Pengxiang Sun

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2025

© Copyright by Pengxiang Sun, 2025

Table of Contents

List of Tables	iv
List of Figures	v
Abstract	vi
List of Abbreviations	vii
Acknowledgements	ix
Chapter 1 Introduction	1
1.1 Introduction to SD-IoT and its threats	1
1.2 Introduction to a framework of dataset generation, DDoS detection and mitigation	2
1.3 Contributions	3
1.4 Outline of the thesis	3
Chapter 2 Background and Literature Survey	5
2.1 Software Defined Network (SDN)	5
2.2 Distributed Denial of Service (DDoS)	6
2.3 Machine Learning Models	7
2.3.1 Decision Tree (DT)	7
2.3.2 Light Gradient Boosting Machine (LightGBM)	8
2.3.3 Extreme Gradient Boosting (XGBoost)	8
2.3.4 Random Forest (RF)	9
2.3.5 Support Vector Machine (SVM)	9
2.3.6 K-Nearest Neighbors (KNN)	9
2.3.7 Naive Bayes (NB)	11
2.3.8 Artificial Neural Networks (ANNs)	12
2.3.9 Logistic Regression	12
2.4 Micro-segmentation	12
2.5 Attribute-based Access Control (ABAC)	13
2.6 Related work	14

Chapter 3	Dataset generation and DDoS detection	27
3.1	Environment of Simulated SDN	27
3.2	Traffic generation	29
3.3	Feature extraction and derivation	33
3.4	Feature analysis	40
3.5	Dataset preprocessing	40
3.6	Evaluation of classifiers	40
3.7	Strategies to optimize performance	43
3.7.1	Feature selection	43
3.7.2	Weighted Majority Vote Ensemble method	45
Chapter 4	DDoS Mitigation	49
4.1	Micro-segmentation	49
4.2	Attribute Based Access Control (ABAC)	52
4.3	Use Case	54
Chapter 5	Conclusion	57
Bibliography	59

List of Tables

Table 2.1	OpenFlow Message Types	7
2.2	Summary of papers in dataset generation	16
2.3	Summary of papers using public dataset	19
2.4	Summary of papers related to DDoS mitigation	23
2.5	Summary of technologies used in the literature for SDN security against DDoS	24
Table 3.1	Source and Destination IPs	29
Table 3.2	Traffic Statistics	33
Table 3.3	Binary classification performance of nine classifiers.	44
Table 3.4	Multi-class classification performance of nine classifiers.	44
Table 3.5	WFI scores for Random Forest (binary classification).	45
Table 3.6	WFI scores for XGB (multi-class classification).	46
Table 3.7	Optimized Classifier Weights for Binary and Multi-class Classi- fication	48
Table 3.8	Performance metrics for majority vote ensemble learning in bi- nary and multi-class classification.	48
Table 4.1	Configuration of ABAC	55

List of Figures

Figure 2.1	Architecture of SDN	6
Figure 2.2	Structure of Decision Tree	8
Figure 3.1	Steps of dataset generation and deployment of DDoS detection	28
Figure 3.2	Topology for generating traffic in SDN	28
Figure 3.3	Process of traffic generation	30
Figure 3.4	Behavior of features during normal and attack	41
Figure 3.5	Evaluation of feature subsets using RFE	47
Figure 4.1	Architecture of proposed micro-segmentation	50
Figure 4.2	Life cycle of micro-segmentation	51
Figure 4.3	Example of micro-segmentation	55

Abstract

Internet of Things (IoT) technology and Software-Defined Networking (SDN) have become key drivers of innovation for several industry applications. Their integration into Software-Defined IoT (SD-IoT) creates new opportunities while introducing significant security challenges. Given the limited resources of many IoT devices, these networks are particularly vulnerable to attacks, with Distributed Denial of Service (DDoS) attacks posing one of the most serious threats to network security and performance. An emerging area of research is the detection and mitigation of DDoS attack on SD-IoT networks.

Many existing DDoS intrusion detection methods rely on outdated datasets, and most of features in these datasets have trivial contribution to attack detection according to the result of feature selection. This thesis presents a framework for DDoS detection and mitigation in SD-IoT. A novel approach is introduced for extracting informative features from network traffic and generating datasets based on those features. In the detection phase, a simulated SDN environment is used to generate normal and DDoS traffic through data-plane algorithms. Traffic features are then extracted to train machine learning models that distinguish between benign and malicious flows. Several classifiers are trained and evaluated based on precision, recall, accuracy, and F1-score, with the weighted majority vote ensemble model selected for its superior performance. For mitigation, a strategy combining micro-segmentation with Attribute-Based Access Control (ABAC) is proposed, enabling effective attack containment and establishing a robust defense-in-depth framework.

The results from our simulated experiments demonstrate the importance of the extracted features in training machine learning models for DDoS detection. Additionally, the weighted majority vote ensemble model shows significant improvements in DDoS detection. A use case further illustrates the efficiency of the proposed micro-segmentation method in mitigating DDoS attacks.

List of Abbreviations

ABAC	Attribute-Based Access Control
ACL	Access Control Lists
ANNs	Artificial Neural Networks
BPNN	Back Propagation Neural Network
DDoS	Distributed Denial of Service
DT	Decision Tree
EFB	Exclusive Feature Bundling
GA	Genetic Algorithm
GBDT	Gradient Boosting Decision Tree
GOSS	Gradient-based One-Side Sampling
GRU	Gated Recurrent Unit
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IoT	Internet of Things
KNN	K-Nearest Neighbors
LightGBM	Light Gradient Boosting Machine
ML	Machine Learning
NB	Naive Bayes
NFV	network function virtualization

NOS	Network Operating System
RF	Random Forest
RBAC	Role Based Access Control
RFE	Recursive Feature Elimination
SD-IoT	Software Defined Internet of Things
SDN	Software Defined Networking
SVM	Support Vector Machine
WFI	Weighted Feature Importance
XGB	Extreme Gradient Boosting

Acknowledgements

I would like to express my deepest gratitude to all those who have supported and guided me throughout the course of my research.

First and foremost, I am sincerely grateful to my supervisor, Dr.Srinivas Sampalli, whose guidance, expertise, and continuous encouragement have been invaluable. His patience, insightful feedback, and unwavering support throughout both Bachelor's and Master's degree have made this thesis possible. I have learned so much under his mentorship, and I am truly thankful for the opportunity to learn from him.

I would also like to express my heartfelt thanks to my parents, for their endless love, support, and belief in me. Their encouragement has been a constant source of strength throughout this journey. I am grateful for their understanding, sacrifices, and for always being there to support me both emotionally and intellectually.

I would also like to thank Dr.Darshana Upadhyay for her patient guidance on my research. And everyone from MyTech lab, I am grateful for their share of experience and ideas, their kindness makes me feel warm.

Chapter 1

Introduction

Internet of Things (IoT) has rapidly evolved into a transformative technology, connecting billions of devices worldwide. It connects devices such as RFID tags, actuators, wearable equipment, unmanned aerial vehicles and other communication devices together to create different kinds of smart ecosystems. Some of the application scenarios are health automation, first responder monitoring and safety system, smart homes and buildings, nifty traffic control and management, industrial control and monitoring system, and so on [62, 74].

The growing complexity and dynamic nature of IoT networks have highlighted the limitations of traditional network architectures. Traditional networks are hard to adapt to the increasing demands for scalability, flexibility, and centralized control, especially when there is a boom growth of connected devices and data generation in IoT schemes. Software Defined Network (SDN) is considered as the solution to the limits mentioned above. It can support IoT networks with rapid evolution and dynamism using programmable planes [62]. In SDN, the control plane is decoupled from forwarding plane and communication between two planes is done through APIs e.g. OpenFlow. SDN is basically layered architecture consist of three layers (1). Data plane, (2). Control plane/controller, and 3). Application layer [8].

1.1 Introduction to SD-IoT and its threats

The integration of Software Defined Networking and the Internet of Things has given rise to Software Defined Internet of Things (SD-IoT). SD-IoT combines the programmability and centralized control of SDN with the interconnected nature of IoT devices, creating a flexible and robust infrastructure for managing IoT networks. This convergence addresses key challenges in IoT environments, such as dynamic device connectivity, heterogeneous communication protocols, and security vulnerabilities, by leveraging the centralized intelligence of SDN controllers.

However, the limited computational resources and battery capacities of IoT devices make SD-IoT more susceptible to certain attacks. Among these, Distributed Denial of Service (DDoS) attacks are particularly concerning, as IoT devices are often easier to compromise. And the phenomenon of forming botnets and launching cyber attacks has become more frequent [34], which underscores the urgent need for effective detection and mitigation mechanisms in SD-IoT networks.

1.2 Introduction to a framework of dataset generation, DDoS detection and mitigation

Both Machine Learning (ML)-based and rule-based Intrusion Detection System (IDS) are security mechanisms that can monitor network traffic or some system activities to distinguish potential security threats. Rule-based IDS requires predefined rules that define patterns of known attack behaviors. In SD-IoT, ML-based IDS provide several advantages over traditional rule-based approaches. ML-based IDS are more adaptable, capable of detecting new and previously unknown attack patterns by learning from network traffic without requiring manual updates. They can perform dynamic behavior analysis, handling complex, multi-stage attacks such as DDoS or advanced persistent threats, which rule-based systems struggle to detect. Additionally, ML models scale efficiently in large SD-IoT environments, reducing false positives and negatives by continuously optimizing classification. In contrast, rule-based IDS rely on static, predefined signatures that are less effective at handling evolving threats and can result in slower detection and higher resource consumption as network traffic grows. Overall, ML-based IDS provide more accurate, real-time detection and response, making them better suited for the dynamic nature of SD-IoT environments.

Public datasets have been crucial in training machine learning models for DDoS detection by providing extensive traffic records and features. However, many of these datasets, such as UNSW-NB15 and KDD-CUP99, are outdated, and research shows that many of their features offer limited value for effective intrusion detection. Consequently, there is a clear need for a method to generate new data and extract informative features tailored to modern intrusion detection.

This thesis proposes an ML-based IDS approach for detecting DDoS attacks in SD-IoT networks. The method extracts eleven key traffic features to classify network

flows. In a simulated SDN environment, normal traffic and DDoS attack traffic are generated using `iperf` and `hping3`, respectively. This dataset is used to train and test 9 machine learning classifiers, including Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Light Gradient Boosting Machine (LightGBM), Extreme Gradient Boosting (XGBoost), Artificial neural networks (ANNs), Naive Bayes (NB), Logistic Regression. Based on four evaluation metrics—accuracy, precision, recall, and F1 score, A weighted majority vote ensemble model is selected as the optimal classifier. The trained ensemble model is then implemented in the control layer of the SDN to monitor the SD-IoT network for potential attacks.

To mitigate detected DDoS attacks, this thesis introduces a novel approach that combines micro-segmentation with dynamic Attribute-Based Access Control (ABAC). By implementing ABAC using Casbin in the controller, the method effectively manages network micro-segments, reducing the attack surface while maintaining robust control and flexibility.

1.3 Contributions

The main contributions of this thesis are as follows:

1. The generation of a traffic dataset in an SD-IoT environment containing both normal and attack traffic, with 12 extracted features for classification.
2. The training and evaluation of multiple machine learning classifiers using the generated dataset, with weighted majority vote ensemble model selected as the best performer based on four evaluation metrics.
3. A novel methodology that integrating micro-segmentation and ABAC to mitigate DDoS attacks in SD-IoT networks, significantly reducing the attack surface while enhancing control and flexibility.

1.4 Outline of the thesis

The remainder of this thesis is organized as follows: Chapter 2 presents the background and literature review. Chapter 3 details the proposed dataset generation and

DDoS detection and their underlying algorithms. Chapter 4 discusses the methodology of the mitigation. Finally, Chapter 5 concludes the thesis and highlights its contributions, and future work to improve the adoptive scalability of this thesis is outlined.

Chapter 2

Background and Literature Survey

This chapter provides the background and surveys the literature on the research related to this thesis. It begins with an introduction to Software-Defined Networking (SDN) and its role in modern IoT networks. Next, it examines the threat of Distributed Denial of Service (DDoS) attacks and explores the use of machine learning techniques for detecting such threats.

The chapter then discusses practical mitigation strategies, focusing on micro-segmentation and attribute-based access control as effective methods against DDoS attacks. It concludes with a review of recent research from the past seven years, highlighting developments in dataset creation, detection, and mitigation strategies for DDoS attacks.

2.1 Software Defined Network (SDN)

Software Defined Networking is a modern architecture that enables centralized network management and programmability by separating the control functions from the data handling processes. Traditional networks rely on various devices—such as routers, switches, firewalls, load balancers, and intrusion detection systems—that each operate with their own protocols and interfaces. This diversity increases complexity and operational costs, particularly as networks grow in size [19].

In SDN, as shown in Figure 2.1, the control and data functions are distinctly separated. The application layer implements essential network services like firewall protection, intrusion detection, routing, and access control. Many controller platforms, including OpenDaylight, ONOS, and Ryu, provide clear northbound interfaces to facilitate communication between the application and control layers. The controller in the control plane collects data from network devices and determines forwarding rules, while the data plane—comprising devices such as switches, routers, and access points—executes these rules based on instructions received through southbound

interfaces. OpenFlow is the primary protocol in southbound interfaces which also make the standard of communication between control plane and data plane. There are 3 types of communication supported by OpenFlow, controller-to-switch, asynchronous and symmetric communication. Controllers can send handshakes messages, switch and flow table configuration to switches in controller-to-switch communication. The asynchronous communications initialized by OpenFlow-compliant switch to send packet-in message, port message, flow message to controller. The symmetric communication can be bidirectional, such as Feature Request/Reply messages [6]. More details of these three communications can be found in table 2.1.

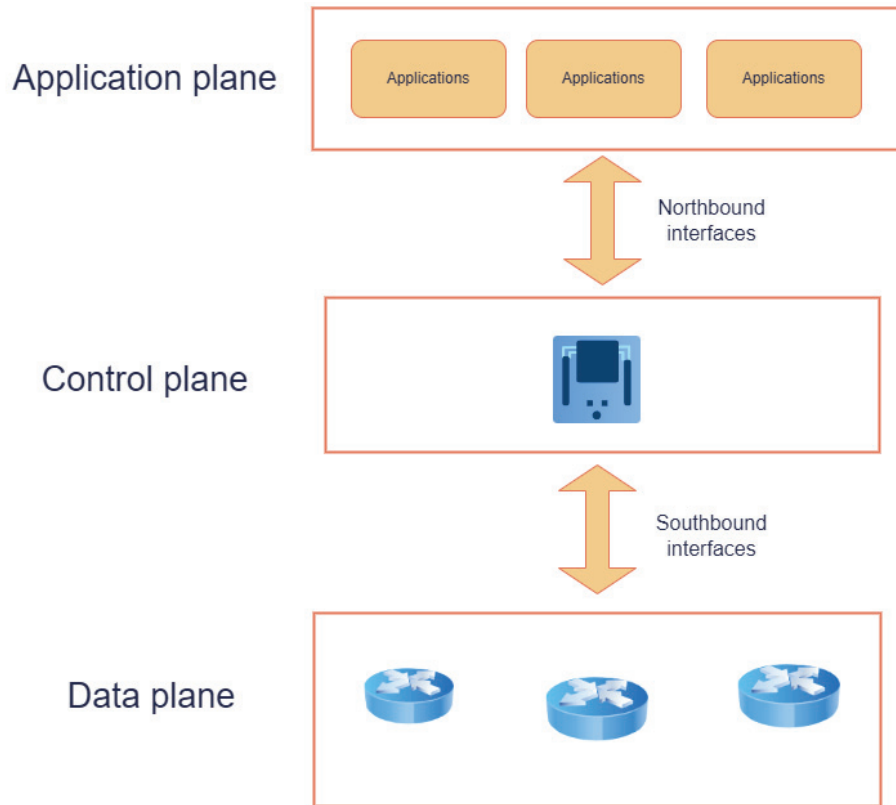


Figure 2.1: Architecture of SDN

2.2 Distributed Denial of Service (DDoS)

Distributed Denial of Service (DDoS) overwhelms the target server with an immense volume of traffic that prevents normal users from accessing the server. [22] It has been one of the most serious threat for network security. It can cause a massive

Table 2.1: OpenFlow Message Types

Category	Messages
Controller-to-Switch Messages	Handshake
	Switch Configuration
	Flow Table Configuration
	Modify State Messages
	Multipart Messages
	Queue Configuration Messages
	Packet-Out Message
	Barrier Message
	Role Request Message
	Set Asynchronous Configuration Message
Asynchronous Messages	Packet-In Message
	Flow Removed Message
	Port Status Message
	Error Message
Symmetric Messages	Hello
	Echo Request
	Echo Reply
	Experimenter

disruption in any information communication technology infrastructure. [5]

2.3 Machine Learning Models

Several Machine Learning (ML) models are adopted in this thesis to detect DDoS attacks in SDN. A well-trained ML classifier can distinguish normal traffic and malicious traffic in network. There are many different kinds of ML algorithms which contribute in classifying abnormal traffic in previous studies. There are 10 ML classifiers are selected in this thesis and they fall into various categories.

A brief overview of classifiers is presented below.

2.3.1 Decision Tree (DT)

Decision tree is a tree-based ML algorithm. It makes classification by forming recursive subsetting of a target field of data based on the most significant features. This

process creates partitions and descendant data subsets, which are also called leaves or nodes. At each level of the tree, the target values within a leaf (or node) become progressively more similar, while those between different leaves (or nodes) become increasingly dissimilar.[13] The structure of Decision tree is shown in Figure 2.2.

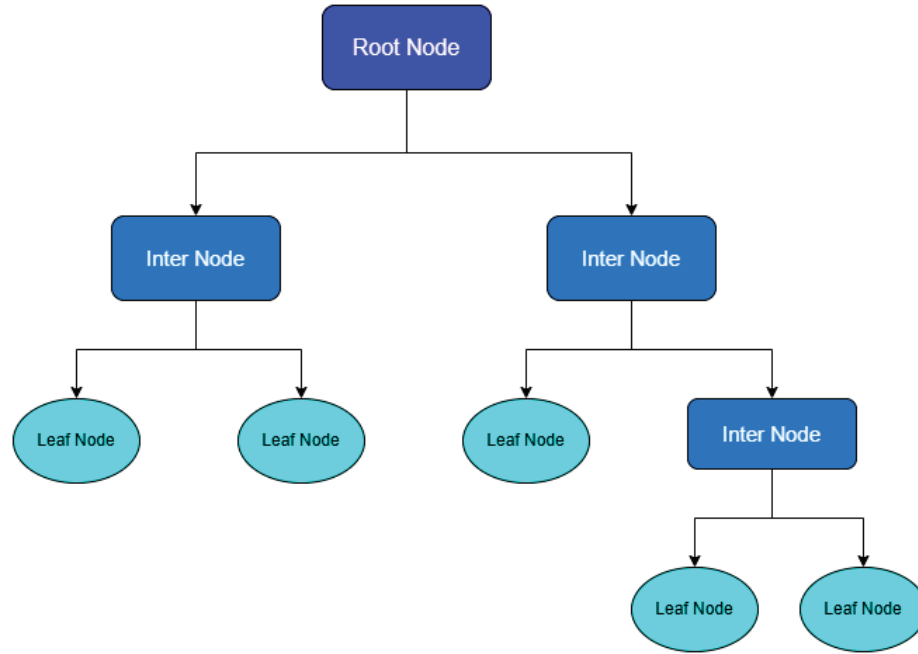


Figure 2.2: Structure of Decision Tree

2.3.2 Light Gradient Boosting Machine (LightGBM)

LightGBM is an advanced gradient boosting algorithm, which builds trees sequentially, each learning from the errors of the previous tree. With the emergence of big data, Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) are implemented in Gradient Boosting Decision Tree (GBDT). This novel GBDT algorithm is called LightGBM.[32]

2.3.3 Extreme Gradient Boosting (XGBoost)

XGBoost is an advanced gradient boosting algorithm based on the Gradient Boosting Decision Tree framework, similar to LightGBM. It introduces several enhancements over traditional gradient boosting, including second-order gradient optimization, regularization techniques, and efficient handling of sparse features. These improvements

enhance its speed and predictive performance, making it a powerful tool for machine learning tasks.

2.3.4 Random Forest (RF)

In Random Forest, the dataset is used to independently train multiple decision tree models. The final prediction is determined by a majority vote among these trees, an ensemble approach that improves accuracy and reduces the risk of overfitting. [7]

2.3.5 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning method applicable to both classification and regression tasks. It enhances predictive accuracy while reducing overfitting. SVM works by identifying the optimal decision boundary that separates different classes in a high-dimensional feature space. For further details, see [29].

2.3.6 K-Nearest Neighbors (KNN)

KNN is an instance-based learning model which can do both classification and regression tasks. KNN is also a type of lazy learning technique (no training phase), because it works by finding k training objects which are closest to the new object, and predict its label based on the predominance of a particular class in this neighborhood. There are some key elements which can have major affect on the performance of KNN: (i) a collection of labeled data points that serve as a reference for classifying new test instances, (ii) distance metrics, (iii) optimal value of k , and (iv) a classification method that assigns a label to the target instance based on the class distribution and distances of its k closest neighbors.[56] Some common distance metrics used in KNN include:

The Euclidean distance between two points in an n -dimensional space:

$$d(A, B) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.1)$$

The Manhattan distance (also known as the L1 norm or taxicab distance) is given by:

$$d(A, B) = |x_2 - x_1| + |y_2 - y_1| \quad (2.2)$$

For an n -dimensional space:

$$d(A, B) = \sum_{i=1}^n |x_i - y_i| \quad (2.3)$$

The Minkowski distance generalizes both Euclidean and Manhattan distances:

$$d(A, B) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}} \quad (2.4)$$

- When $p = 1$, it reduces to the ****Manhattan Distance****. - When $p = 2$, it becomes the ****Euclidean Distance****.

Cosine similarity measures the angle between two vectors:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} \quad (2.5)$$

For vectors A and B :

$$\cos(\theta) = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \cdot \sqrt{\sum_{i=1}^n y_i^2}} \quad (2.6)$$

The corresponding ****Cosine Distance**** is:

$$d(A, B) = 1 - \cos(\theta) \quad (2.7)$$

Definition of Variables:

- **A** and **B** are vectors (or points) in an n -dimensional space.
- n denotes the number of dimensions.
- x_i and y_i are the i -th coordinates of vectors **A** and **B**, respectively, for $i = 1, 2, \dots, n$.
- In Equation (2.2), (x_1, y_1) and (x_2, y_2) represent the coordinates of points **A** and **B** in a two-dimensional space.

- p is a real number parameter in the Minkowski distance that indicates the order of the norm.
- $A \cdot B$ denotes the dot product of vectors \mathbf{A} and \mathbf{B} .
- $\|A\|$ and $\|B\|$ represent the magnitudes (or Euclidean norms) of vectors \mathbf{A} and \mathbf{B} , respectively.
- θ is the angle between vectors \mathbf{A} and \mathbf{B} .

2.3.7 Naive Bayes (NB)

Naive Bayes is a probabilistic classifier based on Bayes' Theorem. Often referred to as the “independence Bayes” classifier, it assumes that features are conditionally independent. Although this assumption may not always reflect real-world data, Naive Bayes often delivers robust performance.

Despite not being the most accurate classifier in every scenario, its simplicity and ease of interpretation make it a popular choice. It is straightforward to construct and easy to understand. Bayes' Theorem is given by:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \quad (2.8)$$

where:

- $P(Y|X)$ is the **posterior probability**, the probability of class Y given the feature set X .
- $P(X|Y)$ is the **likelihood**, the probability of observing X given that the class is Y .
- $P(Y)$ is the **prior probability**, the probability of class Y before observing any features.
- $P(X)$ is the **evidence**, the total probability of X across all possible classes.

2.3.8 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are machine learning models inspired by the structure and functionality of biological neural networks. They consist of an input layer, one or more hidden layers, and an output layer, where each layer is composed of interconnected neurons.[16] ANNs processes data through forward propagation, where weighted inputs are passed through activation functions such as sigmoid, ReLU, and tanh to introduce non-linearity. The network learns patterns through backpropagation, which adjusts the weights using gradient descent or more advanced optimizers like Adam to minimize the loss function.

2.3.9 Logistic Regression

Logistic Regression is a fundamental statistical method used for binary classification tasks, where the goal is to predict the probability that a given input belongs to a particular class. Unlike linear regression, which forecasts continuous outcomes, logistic regression employs the sigmoid function to map a linear combination of input features into a probability value between 0 and 1. This probability is then used to classify the input based on a predetermined threshold, typically 0.5.[26]The model parameters are estimated using maximum likelihood estimation, optimizing a cost function such as the binary cross-entropy loss through gradient descent techniques. Additionally, logistic regression can be adapted for multiclass classification problems via strategies like one-vs-rest or softmax regression. Its simplicity, computational efficiency, and interpretability have led to widespread applications in various domains, including network security, medical diagnosis, and finance, despite its inherent assumption of a linear decision boundary which may limit its performance on more complex, non-linear datasets.

2.4 Micro-segmentation

Micro-segmentation is a network security approach that involves dividing a network into granular, isolated segments, enabling precise control over traffic and reducing the risk of lateral movement when attack is happening. This technique is particularly

relevant in modern, dynamic environments such as data centers and cloud infrastructures, where traditional perimeter-based security measures are no longer sufficient. By segmenting the network into smaller, manageable zones, micro-segmentation allows organizations to implement dynamic security policies for different network segments, enhancing both security and compliance. Moreover, this approach provides improved visibility into network interactions and helps contain potential cyber threats, thereby contributing to a more robust, defense-in-depth security posture. There are four advantages provided by micro-segmentation [17]:

- 1) **Reduced attack surface:** visibility of the whole network is provided by micro-segmentation and innovation and development would not be slowed at the same time.
- 2) **Enhanced breach containment:** Micro-segmentation allows security teams to enforce preset network policies and continuously monitor traffic, which helps limit breaches and speeds up their detection, response, and remediation.
- 3) **Robust regulatory compliance:** Micro-segmentation enables special policies for the segments that needs regulation in the network.
- 4) **Streamlined policy management:** Micro-segmentation makes the management of firewall policies easier.

2.5 Attribute-based Access Control (ABAC)

Attribute-Based Access Control is a dynamic and flexible access control model that grants or restricts user permissions based on attributes associated with users, resources, actions, and environmental conditions. Unlike traditional access control models such as Role-Based Access Control (RBAC), which rely on predefined roles, ABAC enables fine-grained access decisions by evaluating a combination of attributes, such as user identity, device type, location, time of access, and resource sensitivity. This approach enhances security by allowing organizations to enforce context-aware access policies that adapt to changing conditions and minimize unauthorized access risks. ABAC is widely used in modern computing environments, including cloud services, software-defined networking, and zero-trust architectures, where dynamic and scalable access control mechanisms are essential for ensuring data confidentiality, integrity, and regulatory compliance. The logic flow of ABAC can be described as below[28]:

- 1) Operations are initialized by subject to object
- 2) A decision is made by evaluating on Policy, which contains the values of a) Rules, b) Attributes, c) Conditions of Environment.
- 3) Subject can access to object? Depending on the decision.

Where **attributes** are information of subjects, objects and environment. **Subjects** can be users or resources. **Objects** usually are resources under the management of ABAC. **Operation** often includes read, write, edit, delete and so on. **Environment Condition** contains the public variables of system, such as current time, date or security level of system.

2.6 Related work

This section reviews literature on DDoS security published between 2016 and 2024. It summarizes advancements in dataset generation, detection, and mitigation techniques, with a particular focus on machine learning classifiers for DDoS detection and whether the training data is public or self-generated.

Polat et al. [45] simulated an SDN network using a POX controller and OpenVSwitch, employing both hierarchical and star topologies to generate a dataset comprising 12 features and 129,000 records. The dataset, which included normal and attack traffic across TCP, UDP, and ICMP protocols, was refined using sequential forward floating selection and Lasso algorithms to identify 6 key features. Four classifiers—Naive Bayes, SVM, KNN, and ANNs—were evaluated, with KNN achieving the highest performance. Polat et al.[46] also proposed another strategy based on deep learning. They still chose POX controller in simulated network. The dataset used to train SSAE+SoftMax 4-layer deep network model contained more features and less rows (42 features and 17779 rows) compared the dataset in their other work which is mentioned above.

Ahuja et al. [2] generated a dataset with 104,345 records and 23 features, later reducing it to 8 significant features through feature extraction. Their network simulation utilized a Ryu controller and Mininet, while Mgen and Hping3 produced normal and attack traffic. Similarly, Ye et al. [73] employed the Floodlight controller to create a dataset containing 6 features and 30,000 records, with SVM emerging as the best

classifier. Myint Oo et al. [41] developed their own dataset—comprising normal traffic, SYN floods, and UDP floods—using an OpenDaylight controller. They extracted 5 features and customized an advanced SVM model that outperformed alternative classifiers.

Wang et al. [71] secured an SDN against DDoS attacks by employing multiple controllers. Their simulation used a Ryu controller, with D-ITG and Hping3 generating both normal and attack traffic, and 4 vector tuples were extracted from the traffic. The Back Propagation Neural Network (BPNN) delivered the best results, and their strategy included activating backup controllers when the primary controller was under attack.

Majd Latah and Levent Toker [35] created a relatively small dataset (8 features and 960 rows) to train Multi-Layer Perceptron model. POX controller was deployed in a tree-topology simulated SDN. They used D-ITG and Scapy to collect normal and attack traffic. Fan et al.[18] proposed a novel machine learning model called RF-SVM-IL to detect DDoS in SDN. They integrated Random Forest, SVM and Incremental Learning. Their simulated network used Ryu controller to form a dataset. Wang et al.[70] utilized Floodlight controller and mininet to simulate SDN. The controller, switches and hosts were arranged following hierarchical tree topology. Scapy library were used to generate normal traffic, SYN flood, UDP flood and ICMP flood. The dataset contained 6 columns and 35,000 rows. Rahman et al.[48] used Ryu controller and Mininet to create SDN. Hping3 were chosen to produce SYN flood, UDP flood, ICMP flood and Botnet. They created a dataset containing 24 columns and 326232 rows and this dataset was used to train J48 model. Table 2.2 shows the summary of papers related in dataset generation.

Table 2.2: Summary of papers in dataset generation

Ref No	Source of dataset	Size of dataset (Number of Columns x Number of records)	Tools and Best classifier	Topology	Size of network	Method of feature selection	Type of attack
Polat et al. (2020)[45]	Self-generated	12 x 129000 = 1548000	Controller: Pox - and hping3 KNN	hierarchical and star topologies	Controller(pox):1 Switch (OVS and vm):3 Host:6	The Relief algorithm The sequential forward floating selection algorithm The Lasso algorithm	DDoS: SYN flood UDP flood ICMP flood
Ahuja et al. (2021)[2]	Self-generated	104345x23 = 2399935 (23 are extracted, 16 are used to train, only 8 are significant)	mgen and hping3 SVM-RF	Tree topology Emulator: Mininet	Controller (Ryu):1 Switch(ovs): 9 Hosts: 13-15	None	DDoS
Ye et al. (2018)[73]	Self-generated	6 x 30000 = 180000	NA and hping3 SVM	Star topology Emulator: Mininet	Controller (Floodlight): 1 Switch (OFS):5 Hosts:5	None	DDoS: SYN flood UDP flood ICMP flood
Myint Oo et al. (2019)[41]	Self-generated	5 x None	None and Scapy customized SVM	hierarchical SDN topology with a controller cluster Emulator: Mininet	Controller (ODL):3 Switch (OFS):4 Host:4	None	DDoS: SYN flood UDP flood
Wang et al. (2020)[70]	Self-generated	6 x 35000 = 210000	Scapy library CNN-3C2P2F	hierarchical tree topology Emulator: Mininet	Controller (Floodlight): 1 Switch (OVS):6 Hosts:8	None	DDoS: SYN flood UDP flood ICMP flood
Polat et al. (2020)[46]	Self-generated	42 x 17779 = 746718	hping3 SSAE SoftMax 4-layer deep network model	Hierarchical and Star-Like Local Topology Emulator: SUMO	Controller(pox):3 Switch (RSU):6 Host:100	None	DDoS: SYN flood UDP flood ICMP flood
Latah and Toker (2018)[35]	Self-generated	8 x 960 = 7680	: D-ITG and Scapy MLP	tree topology Emulator: Mininet	Controller(pox):1 Switch (OFS):4 Host:9	None	DDoS: SYN flood UDP flood ICMP flood
Fan et al. (2021)[18]	Self-generated	5 x None	Hping3 RF-SVM-IL,	hierarchical SDN topology Emulator: Mininet	Controller (Ryu):1 Switch(ovs): 3 Hosts: 3	None	DDoS: SYN flood UDP flood ICMP flood
Wang, et al. (2019)[71]	Self-generated	4 x None	D-ITG and Hping3 BPNN	hierarchical SDN topology with a controller cluster Emulator: Mininet	Controller (Ryu):5 Switch(ofs): 10 Hosts: 4	None	DDoS: SYN flood UDP flood ICMP flood Low-traffic novel DDoS attacks (targeting the SDN controller)

Ref No	Source of dataset	Size of dataset (Number of Columns x Number of records)	Tools and Best classifier	Topology	Size of network	Method of feature selection	Type of attack
Rahman, et al. (2019)[48]	Self-generated	24 x 326232 = 7829568	None and Hping3 J48	star topology Emulator: Mininet	Controller (Ryu):1 Switch(ofs): 1 Hosts: 5	None	DDoS: SYN flood UDP flood ICMP flood Botnet
My proposed work	Self-generated	12 x 140000 = 1680000	Iperf3 and hping3 Weighted Majority Vote Ensemble	Tree topology	Controller (Ryu):1 Switch(ofs): 6 Hosts: 18	WFI and RFE	DDoS: SYN flood UDP flood ICMP flood IP spoofing FIN flood Botnet

In addition to self-generated dataset, there are some public datasets related to SDN which contribute to field of DDoS detection, such as CICDDoS 2019, CICDDoS 2017, KDD99CUP, NSL-KDD, UNSW-NB15, CIC-IDS 2018, CAIDADDOS and so on. Kareem and Jasim [31] proposed a novel algorithm to detect DDoS. It was a partial decision tree algorithm called PART. Several classifiers were compared with PART, such as RT, REPT and RF. PART performed the best and achieved 99.77% of accuracy. Rehman et al.[67] tested the performance of machine learning algorithms on detecting reflection DDoS attack and exploitation attack. They had selected a small part of dataset CICDDoS2019 to train classifiers. The result showed 99.69% accuracy in detecting reflection attacks and 99.94% accuracy for classifying exploitation using Gated Recurrent Unit (GRU). In order to overcome the emergence of increasing data size and growth of data dimensions, Shen et al.[54] proposed an ensemble method based on Extreme Learning Machine. Bat algorithm was used as an ensemble pruning method. 99.3% of accuracy was resulted. Shone et al.[55] used Random Forest to detect DDoS. In order to improve the accuracy and execution time, Non-Symmetric Deep Auto Encoder was applied to the training dataset to utilize RF. Gao et al.[21] proposed a strategy of DDoS detection. A voting system and ensemble ML models were used and DT, RF, KNN, DNN were selected as the basic classifiers. Chen et al.[10] had tested the performance of multiple classifiers using CUP99 as training dataset. XGBoost outperformed SVM, GBDT and RF with false-positive rate of 0.008. Perez-Diaz et al.[44] proposed a work to protect SDN controller from DDoS by indentifying low-rate DDoS attack. Multiple classifiers are tested, such as J48,

MLP, SVM, RF, REPTree and RT. The result showed a accuracy of 95% in detecting low-rate DDoS attack. A hybrid CNN-ELM model was introduced by Wang and Wang et al.[69]. The result showed accuracies of 98.92% and 99.91% on CIC-IDS2017 and InSDN dataset receptively. A source-based DDoS detection method was presented by Priyadarshini and Barik [47]. They integrated two datasets to train LSTM model, CTU-13 provided malicious traffic and IXCS-2012 provided normal traffic. The accuracy in their work was 98.99%.

Good use of feature selection can contribute to the performance of classifiers in DDoS detection. Latah and Toker [35] applied PCA to reduce the number of features from 41 to 9 for NSL-KDD dataset. Chi-Square test feature selection was applied on CIC-IDS 2018 by luong et al.[38] They reduce the feature number from 84 to 67. The dataset after feature selection was applied on SVM, NB, DT, RF and Deep Neural Network. SVM and DNN performed the best. Tayfour and Marsono [61] used four public datasets to train classifiers (InSDN2020, CICIDS2017, NSL-KDD and UNSW-NB15). They proposed an ensemble classifier including NB, KNN, DT and ET to detect DDoS. ET resulted True-Positive Rate of 0.985 and Fasle-Positive Rate of 0.008. In considerations of the number of features in total of these four datasets, feature selection methods may have a further contribution to the result.

In addition to the accuracy of classifiers, time effectiveness and power consumption are also important for DDoS detection. Shafi et al.[52] showed a work of using MLP, Alternate DT and RNN to detect DDoS attack. The result showed 0 packet delay for 1000 packets compared to cloud network.

Although some datasets used in previous studies—such as DARPA1998 and KDD-CUP99 from the 1990s—are outdated, they continue to inspire innovative approaches to DDoS detection using machine learning. Table 2.3 summarizes studies that employ public datasets to train these models.

Table 2.3: Summary of papers using public dataset

Title	Dataset	Methodology	Algorithm	Pros	Cons	Performance Metrics
Kareem and Jasim. (2022) [31]	CICDDOS2019	ML	Partial DT	High accuracy: 99.77%	Network is low-scaled (2 switches, 10 hosts), compared to only tree-based algorithms	Accuracy, precision, recall, F1
Rehman, et al. (2021)[67]	CICDDoS2019	ML	GRU, RNN, SMO, NB	High accuracy: 99.69%	Small test and training data size	Accuracy, precision, recall, F1
Tuan, et al. (2020) [66]	KDD99CUP, UNBS-NB 15	ML	SVM, ANNs, NB, DT, USML	High accuracy: 98.08%, FAR: 1.92%	KDD99CUP dataset outdated	Accuracy, FAR, sensitivity, specificity, FPR, AUC, MCC
Shen et al. (2018) [54]	NSL-KDD, KDDCup99	Ensembled ML	ELM with BAT algorithm	Performs well in an ensemble setting	Datasets are outdated	Accuracy, precision, recall, F1
Shone et al. (2018) [55]	NSL-KDD, CICIDS2017	DL	RF-based on SDAE	Lowers model complexity	Datasets outdated	Time, accuracy
Gao et al. (2019) [21]	NSL-KDD	Ensembled ML	DT, RF, KNN, DNN with a voting mechanism	Voting system selects the best classifier	Dataset outdated	Accuracy, precision, recall, F1
Chen et al. (2018) [10]	KDDCUP99	ML	XGBoost	FPR is low (0.008); Compared with SVM, GBDT, RF	KDDCUP99 dataset outdated	Accuracy, FPR, training time
Gao et al. (2018)[20]	DARPAIDS	DL	Bayesian network	Detects packet-in flooding attack with low overhead	Dataset outdated (1999)	CPU usage, accuracy

Title	Dataset	Methodology	Algorithm	Pros	Cons	Performance Metrics
Latah and Toker (2018) [35]	NSL-KDD	ML	DT, ELM, NB, LDA, NN, SVM, RF, KNN, AdaBoost, RUSBoost, LogitBoost, BaggingTrees	PCA reduces features from 41 to 9	KDD99CUP dataset out-dated	Accuracy, precision, recall, F1, execution time, McNemar's test
Shafi et al. (2019) [52]	UNSW-NB15	ML	MLP, RNN, DT	0 packet delay compared to cloud network	No performance evaluation for classifiers	Network delay, throughput, fairness
Cui et al. (2019) [12]	CAIDADDoS	ML	SVM	100% detection rate, low FPR	Dataset out-dated (2007)	Detection rate, FPR
Tuan et al. (2019) [65]	CAIDADDoS	ML	KNN	Guarantees device capacity	Limited to TCP-SYN flood	Accuracy, precision, recall, F1
Perez-Diaz et al. (2020) [44]	CICDoS2017	ML	J48, MLP, SVM, RF, REPTree, RT	Tested multiple classifiers	Only detects low-rate DDoS, 95% accuracy	Accuracy, precision, recall, F1, FAR
Luong et al. (2020) [38]	CIC-IDS2018	ML	SVM, NB, DT, RF, DNN	ML and DL classifiers tested; Chi-square test for feature selection	DT cannot detect abnormal traffic	Accuracy, precision, recall, F1
Khedr et al. (2023) [33]	Edge-IIoTset	ML	SVM, GNB, KNN, RF, DT, Binomial LR	High accuracy (99.79%)	Limited to TCP flood	Accuracy, recall, F1
Revathi et al. (2021) [49]	KDDCUP99	ML	SVM	SMCA reduces features from 41 to 9	KDDCUP99 dataset out-dated	Accuracy, precision, recall, F1
Tayfour and Marsono (2021) [61]	InSDN2020, CICIDS2017, NSL-KDD, UNSW-NB15	Ensembled ML	Ensemble of NB, KNN, DT, ET	RSMQ minimizes controller load	Outdated datasets and Feature selection can be used on 4 datasets to improve performance	Accuracy, precision, recall, F1, TPR, FPR

Title	Dataset	Methodology	Algorithm	Pros	Cons	Performance Metrics
Banerjee and Chakraborty (2021) [4]	Kaggle dataset	ML	NB, KNN, K-means, Linear Regression	Tested multiple classifiers	Lack of performance metrics	Efficiency
Yungaicela-Naula et al. (2021) [75]	CIC-DoS2017, CIC-DDoS2019	ML and DL	SVM, KNN, RF	PCA reduces features from 76 to 15	Lower accuracy for application-layer DDoS (95%)	Accuracy, precision, recall, F1
Xu et al. (2019) [72]	NSL-KDD	ML	K-FKNN	Combines KMeans++ and KNN	Outdated dataset; Long detection time	Precision, Recall, F-measure, Detection time
Zhao et al. (2021) [76]	DDoSAttack2007	ML	SOM	Fast processing time; High detection accuracy	Dataset outdated	Accuracy, Processing time
Deepa et al. (2019) [14]	CAIDA	Ensembled ML	KNN-SOM, NB-SOM, SVM-SOM	SVM-SOM accuracy: 98.14%	Detection rate (95%) could improve with feature selection	Accuracy, detection rate, FAR
Tan et al. (2020) [60]	NSL-KDD	Ensembled ML	K-Means-KNN	High precision (99.03%)	Dataset outdated	Accuracy, precision, recall, FAR
Swami et al. (2020) [58]	UNSW-NB15, CICIDS2017, NSL-KDD	Ensembled ML	Voting-CMN, Voting-RKM, Voting-CKM	High accuracy (99.68%)	No feature selection applied	Accuracy, precision, recall, F1
Wang and Wang (2022) [69]	CIC-IDS2017, InSDN	Ensembled ML	CNN-ELM	High accuracy (98.92%, 99.91%)	No feature selection applied	Accuracy
Haider et al. (2019) [25]	CICDDoS2017	DL	CNN	High accuracy (99.48%)	No real-world/simulated testing	Accuracy, precision, recall, F1
Priyadarshini and Barik (2019) [47]	CTU-13, ISCX-2012	DL	LSTM	Propose source-based DDoS detection	Only one performance metric	Accuracy
Liang and Znati (2019) [36]	CICIDS2017	DL	LSTM	High precision (99%)	No feature engineering	Precision, Recall, F-measure

Title	Dataset	Methodology	Algorithm	Pros	Cons	Performance Metrics
Novaes et al. (2021) [42]	CICDDoS2019	DL	Adversarial Deep Learning (GAN)	Proposes detection and mitigation strategy	No feature engineering	Accuracy, Precision, Recall, F-measure
My proposed work	self-generated	ML	RF, DT, SVM, KNN, NB, LR, ANNs, LGB, XGB	1)High accuracy 2)Detection and mitigation strategy 3)Various kinds of classifiers are tested.	Performance is not good enough for classifying multi-class values	Accuracy, Precision, Recall, F-measure

Despite the use of DDoS detection in network security, the use and contribution of DDoS mitigation can not be ignored. Shashidhara et al.[53] introduced a mitigation method called SDN-chain, which was deployed in application layer in SDN. A consensus protocol was used for SDN security and reliability. Additionally, Cryptographic primitives was used to design a security protocol. This security protocol also was analyzed by details. The results showed that SDN-Chain possessed characteristics such as security, efficiency, a reduced tendency to centralize, and compatibility with wireless environments where resources are limited.

Honeypot is a popular and efficient strategy against DDoS. According to the work of Sumadi et al.[57], Representational State Transfer API can be used to prevent the attack by conveying the flow rule modification message. The result showed that, the mitigation process can be triggered in 31-49 ms once the attack is detected. Tian et al.[64] proposed a work of integrating dynamic-based honeypot and game model to protect Industrial Internet of Things (IIoT) from DDoS. They employed the Prelec weighting function to capture limited rationality and proposed a DBHM to quantify the interactions between the attacker and defender. They then applied the RDs criteria to classify and identify stable equilibrium strategies.

Network slicing can be utilized to prevent mobile communications from DDoS attack by dividing network resources into distinct slices. It also makes the network management easier. Sattar et al.[51] deployed network slicing in SDN to mitigate

DDoS. A mathematical method was introduced by the authors to guarantee the end-to-end latency in each network slice. Kabdjou et al.[30] emerged multiple technologies such as deception, network slicing, SDN, network function virtualization (NFV) and decision-making system, to build secure and manageable network environment. Additionally, the proposed architecture consistently upholds quality of service standards while integrating a crucial filtering mechanism to mitigate potential security risks. Thantharate et al.[63] introduced a 5G architecture which can classify and assign incoming traffic to its proper network slice. The users or resources in the network are allowed to communicate only after validation and permit. Vishwakarma and Jain [68] used machine learning detection to construct a honeypot-based strategy. The decoy system would redirect the malicious traffic as soon as they were detected.

Abou El Houda et al.[1] proposed SDNWisdom to protect OpenFlow switches from DNS amplification attack. It mapped requests and responses of DNS to shield victims and resources. Snort is rule-based system with buzzing alarm of DDoS attack, it was deployed by Manso et al.[39] In their proposed work, DDoS attack can be mitigated after 3.07s on average with 0% of packet loss. Ali and Yousaf [3] introduced a three-tier mitigation approach for resource consumption attack. Their work focused on traffic load, throughput and failure rate. IoT, packets and queues are validated in Tier 1, 2, 3 respectively. In table 2.4, the papers about attack mitigation are concluded and listed.

Table 2.4: Summary of papers related to DDoS mitigation

Title	Strategy	Deployment Layer	Target Attack	Scale of Network
Shashidhara, et al. 2021 [53]	Blockchain	Application layer	IP spoofing, Modification attacks, DDoS attacks, Insider attacks	Low
Sumadi, et al. 2022 [57]	Honeypot	Data layer	ICMP flood, TCP flood	Medium to large
Tian, et al. 2021 [64]	Honeypot	Data layer	APT attack	Medium to high
Sattar, et al. 2019 [51]	Network slicing	Control layer	DDoS attacks and slicing-initiated attacks	Large

Title	Strategy	Deployment Layer	Target Attack	Scale of Network
Kabdjou et al. 2024 [30]	Network slicing	Control layer	Botnet and DDoS attacks	Large
Thantharate, et al. 2020 [63]	Network slicing	Control layer	Botnet and DDoS attacks	Large
Vishwakarma and Jain 2019 [68]	Honeypot	Data layer	Zero-day DDoS attack and botnet	Medium
Abou El Houda et al. 2020 [1]	Access control	Data layer	DNS amplification	Medium to large
Manso et al. (2019) [39]	Access control	Control layer and data layer	DDoS attacks	Medium
Ali and Yousaf. 2020 [3]	Access control	All layers	Replay attack, MiMA attack, forgery attack, and DDoS attack	Medium to large
My proposed work	Hybrid – network slicing (micro-segmentation) and access control (ABAC)	Control layer	DDoS attacks and Botnet	Medium

Table 2.5 provides a summary of the literature reviewed in this chapter, organized by dataset generation, attack detection, and attack mitigation. Additionally, it includes an overview of this thesis’s contributions, clearly highlighting its novel aspects.

Table 2.5: Summary of technologies used in the literature for SDN security against DDoS

	Data generation	Data generation	Data generation	Attack detection	Attack detection	Attack detection	Attack mitigation	Attack mitigation
Title	Ryu	Tree topology	Multiple attack types	Multiple metrics	Comparison of classifiers	Self-generated dataset	Hybrid strategy	Multiple target attacks
Polat et al. (2020)	X	X	✓	✓	✓	✓	X	X
Ahuja et al. (2021)	✓	✓	✓	✓	✓	✓	X	X
Ye et al. (2018)	X	X	✓	✓	✓	✓	X	X
Myint Oo et al. (2019)	X	X	✓	✓	✓	✓	X	X

Wang et al. (2020)	X	✓	✓	✓	✓	✓	X	X
Polat, et al. (2020)	X	X	✓	✓	✓	✓	X	X
Latah and Toker. (2018)	X	✓	✓	✓	✓	✓	X	X
Fan, et al. (2021)	✓	X	✓	✓	✓	✓	X	X
Wang, et al. (2019)	✓	X	✓	✓	✓	✓	X	X
Rahman, et al. (2019)	✓	X	✓	✓	✓	✓	X	X
Karrem and Jasim. (2022)	X	X	X	✓	X	✓	X	X
Rehman, et al. (2021)	X	X	X	✓	✓	✓	X	X
Tuan, et al. (2020)	X	X	X	✓	✓	X	X	X
Shen et al. (2018)	X	X	X	✓	✓	X	X	X
Shone et al.	X	X	X	✓	X	X	X	X
Gao et al. (2019)	X	X	X	✓	✓	X	X	X
Chen et al. (2018)	X	X	X	✓	X	X	X	X
Gao et al. (2018)	X	X	X	✓	✓	X	X	X
Latah and Toker. (2018)	X	X	X	✓	✓	X	X	X
Shafi et al. (2019)	X	X	X	✓	✓	X	X	✓
Cui et al. (2019)	X	X	X	✓	✓	X	X	X
Tuan et al. (2019)	X	X	X	✓	✓	X	X	✓
Perez-Diaz et al. (2020)	X	X	X	✓	✓	X	X	✓
Luong et al. (2020)	X	X	X	✓	✓	X	X	✓
Khedr et al. (2023)	X	X	X	✓	✓	X	X	✓
Revathi et al. (2021)	X	X	X	✓	✓	X	X	✓
Tayfour and Marsono (2021)	X	X	X	✓	✓	X	X	✓
Banerjee and Chakraborty. (2021)	X	X	X	X	✓	X	X	✓
Yungaicela-Naula et al. (2021)	X	X	X	✓	✓	X	X	✓
Xu et al. (2019)	X	X	X	✓	X	X	X	✓

Zhao et al. (2021)	X	X	X	✓	✓	X	X	X
Deepa et al. (2019)	X	X	X	✓	✓	X	X	X
Tan et al. (2020)	X	X	X	✓	X	X	X	X
Swami et al. (2020)	X	X	X	✓	✓	X	X	X
Wang and Wang. (2022)	X	X	X	X	✓	X	X	✓
Haider et al. (2019)	X	X	X	✓	✓	X	X	X
Priyadarshini and Barik. (2019)	X	X	X	X	✓	X	X	X
Liang and Znati (2019)	X	X	X	✓	✓	X	X	X
Novaes et al. (2021)	X	X	X	✓	✓	X	X	✓
Shashidhara, et al. 2021	X	X	X	X	X	X	X	✓
Sumadi, et al. 2022	X	X	X	✓	✓	X	X	X
Tian, et al. 2021	X	X	X	✓	✓	✓	X	X
Sattar, et al. 2019	X	X	X	X	X	X	X	✓
Kabdjou et. al. 2023	X	X	X	✓	✓	X	X	✓
Thantharate, et al. 2020	X	X	X	✓	✓	✓	X	✓
Vishwakarma and Jain 2019	X	X	X	✓	✓	X	X	✓
Abou El Houda et al. 2020	X	X	X	✓	✓	X	X	X
Manso et al. (2019)	X	X	X	✓	✓	✓	X	X
Ali and Yousaf. 2020	X	X	X	X	X	X	X	✓
My proposed work	✓	✓	✓	✓	✓	✓	✓	✓

Chapter 3

Dataset generation and DDoS detection

In this chapter, we outline the process for creating the dataset and implementing the DDoS detection strategy. The key steps are as follows:

1. Build a simulated SDN environment.
2. Generate both normal and attack traffic within this environment.
3. Extract features from the traffic, starting with 17 raw features and deriving 11 informative ones.
4. Create and preprocess the dataset.
5. Train the classifiers introduced in Chapter 2 using the generated dataset.
6. Evaluate classifier performance using metrics such as precision, accuracy, recall, and F1-score.
7. Enhance classifier performance through techniques like feature selection and a majority vote ensemble method.
8. Deploy the best-performing classifier into the SDN controller.

Figure 3.1 illustrates these steps.

3.1 Environment of Simulated SDN

The simulated SDN is deployed on a virtual environment running Linux Ubuntu 20.04 on VirtualBox, with an Intel i5-12400KF CPU and 12 GB of RAM. Its architecture comprises a controller, OpenFlow-compatible switches, and end devices. Mininet is used as the emulator for its lightweight and flexible design, providing reliable OpenFlow switch emulation. The controller, also known as the Network Operating System (NOS), centrally manages the network. Common controllers include NOX [23], POX [43], Ryu [11], ONOS, OpenDaylight, and Floodlight; Ryu is particularly popular due to its active developer community and Python-based framework that integrates well with Mininet. Further details of controllers can be found in [77].

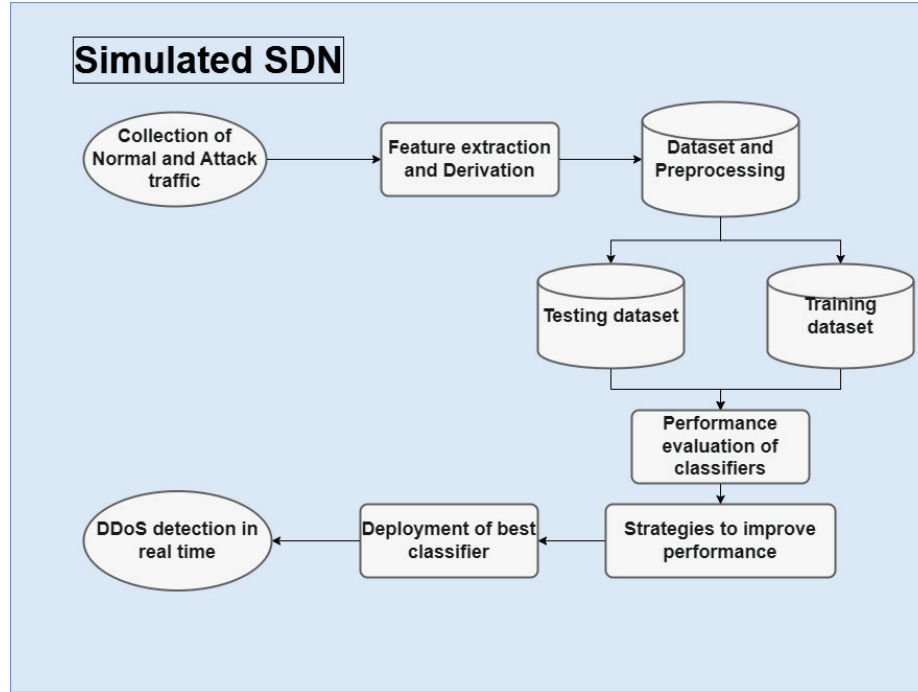


Figure 3.1: Steps of dataset generation and deployment of DDoS detection

Normal and attack traffic are generated within the topology illustrated in Figure 3.2. The simulated SDN consists of one Ryu controller, six switches (S1–S6), and 18 hosts (h1–h18), with each switch connecting to three hosts and the Ryu controller linked to all switches.

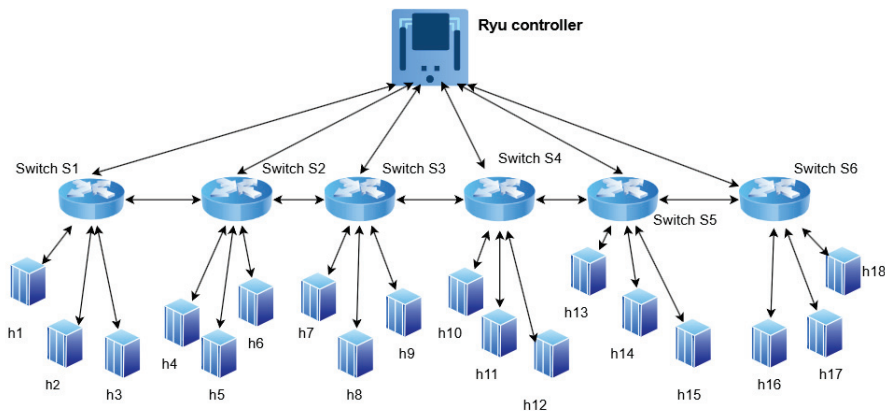


Figure 3.2: Topology for generating traffic in SDN

3.2 Traffic generation

Normal and DDoS attack traffic are generated within the simulated network using Iperf [27] and hping3. Three types of normal traffic—ICMP, TCP, and UDP—are produced: ICMP traffic is generated using the ping command in Mininet, while TCP and UDP traffic are created with Iperf across all hosts.

Five types of attack traffic are generated with hping3, including SYN flood, UDP flood, ICMP flood, botnet-based, and FIN flood attacks. In the simulated attack scenario, one host is compromised to launch a DDoS attack against a randomly selected victim, with IP spoofing employed to disguise the source. As shown in Table 3.1, destination IP addresses range from 10.0.0.1 to 10.0.0.18, corresponding to the 18 hosts in the network, while the source IP addresses are spoofed.

Table 3.1: Source and Destination IPs

Src IPs	Dest IPs
10.0.0.1	10.0.0.1
10.0.0.16	10.0.0.10
10.0.0.17	10.0.0.11
0.11.224.97	10.0.0.12
10.0.0.18	10.0.0.13
10.0.0.13	10.0.0.14
10.0.0.2	10.0.0.15
10.0.0.3	10.0.0.16
10.0.0.4	10.0.0.17
10.0.0.5	10.0.0.18
...	10.0.0.2
0.131.198.43	10.0.0.3
0.131.44.44	10.0.0.4
35.154.208.22	10.0.0.5
238.195.39.221	10.0.0.6
2.245.100.11	10.0.0.7
172.11.211.34	10.0.0.8
190.68.47.37	10.0.0.9

The traffic generation process is divided into three phases, as illustrated in Figure 3.3. Initially, normal traffic is generated for a period T1 using Algorithm 1. This

is followed by an attack phase lasting T_2 , during which Algorithm 2 is used to produce attack traffic. Finally, normal traffic resumes for a period T_3 . This approach is necessary because the traffic features used to identify malicious activity are based on the flow table statistics of each switch. After a DDoS attack, flow tables continue to record attack flows until these entries expire according to the ‘idle_timeout’ or ‘hard_timeout’ settings. Consequently, while T_1 and T_2 can be variable, T_3 must be longer than the ‘idle_timeout’ and ‘hard_timeout’—set at 20 and 120 seconds, respectively—to ensure that the attack has fully cleared. ‘Idle_timeout’ refers to the number of seconds a flow entry can stay in the flow table without matching any packets. And ‘hard_timeout’ means the maximum number of seconds a flow entry is allowed to stay in the flow table regardless of hits or miss.

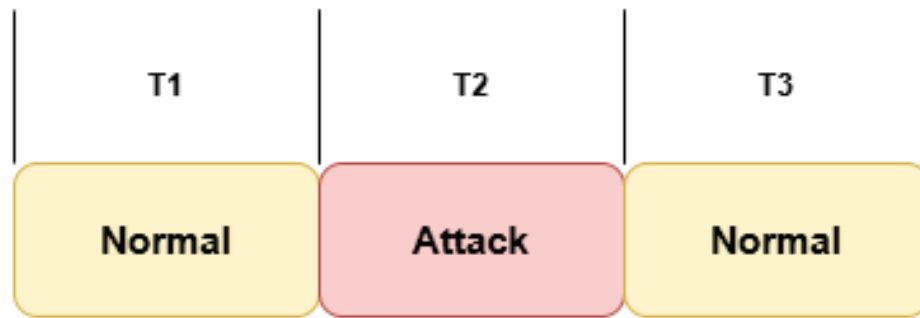


Figure 3.3: Process of traffic generation

Algorithm 1: Traffic Generation Algorithm

Input: $h1, h2, \dots, hn$: List of hosts in the network

w, W : Range of waiting time

Output: *Normal traffic generation*

```

1 while True do
2   Hosts  $\leftarrow$  [h1, h2, ..., hn]
3   Total_wait_time  $\leftarrow$  0
4   foreach host h in Hosts do
5     Wait_time  $\leftarrow$  randint( $w, W$ )
6     Total_wait_time += Wait_time
7     Dst_host  $\leftarrow$  choice(Hosts)
8     while Dst_host == h do
9       Dst_host  $\leftarrow$  choice(Hosts)
10    Protocol  $\leftarrow$  choice([ICMP, TCP, UDP])
11    if Protocol == ICMP then
12      h.cmd("ping {Dst_host} -c 100 &")
13    else if Protocol == TCP then
14      h.cmd("iperf -p 5050 -t {Wait_time} -c {Dst_host}")
15    else if Protocol == UDP then
16      h.cmd("iperf -p 5051 -t {Wait_time} -u -c {Dst_host}")
17    sleep(Total_wait_time)

```

Algorithm 2: Attack Traffic Generation

Input: $h1, h2, \dots, hn$: List of hosts in the network

$Attack_types$: List of different types of DDoS

w, W : Range of waiting time

Output: *Attack traffic generation*

```

1 while True do
2   Hosts  $\leftarrow$  [h1, h2, ..., hn]
3   Wait_time  $\leftarrow$  randint( $w, W$ )
4   Src, Dst  $\leftarrow$  sample(Hosts, 2)
5   Attack  $\leftarrow$  choice(Attack_types)
6   if Attack == "icmp" then
7     Src.cmd("timeout 20s hping3 -1 -V -d 120 -w 64 -p 80 -rand-source
8       -flood {Dst}")
9     sleep(Wait_time)
10  else if Attack == "udp" then
11    Src.cmd("timeout 20s hping3 -2 -V -d 120 -w 64 -rand-source -flood
12      {Dst}")
13    sleep(Wait_time)
14  else if Attack == "syn" then
15    Src.cmd("timeout 20s hping3 -S -V -d 120 -w 64 -rand-source -flood
16      {Dst}")
17    sleep(Wait_time)
18  else if Attack == "botnet" then
19    Src.cmd("timeout 20s hping3 -1 -V -d 120 -w 64 -flood -a {Dst}
20      {Dst}")
21    sleep(Wait_time)
22  else if Attack == "Fin" then
23    Src.cmd("timeout 20s hping3 -F -V -d 120 -w 64 -flood {Dst}")
24    sleep(Wait_time)

```

3.3 Feature extraction and derivation

A Python script integrated into the Ryu controller collects traffic statistics from both normal and attack scenarios using 17 predefined metrics via Ryu’s APIs [50]. Table 3.2 details these statistics along with their descriptions and examples.

Table 3.2: Traffic Statistics

Name	Description	Example
timestamp	Time of flow	1589932746
datapath_id	ID of Datapath (Open-FlowSwitch)	“1”
flow_id	ID of flow	10.0.0.104380410.0.0.1450501
ip_src	Source IP of flow	10.0.0.1
tp_src	Source port number of flow	5050
ip_dst	Destination IP of flow	10.0.0.14
tp_dst	Destination port number of flow	5051
ip_proto	Protocol	1
icmp_code	ICMP code	5
icmp_type	ICMP type	6
flow_duration_sec	Time flow has been alive in seconds	4
flow_duration_nsec	Time flow has been alive in nanoseconds beyond duration_sec	480000000
idle_timeout	Number of seconds idle before expiration	20
hard_timeout	Number of seconds before expiration	120
flags	Bitmap of OFPFF_* flags	1
packet_count	Number of packets in flow	50776
byte_count	Number of bytes in flow	3351216
lookup_count	How many times packets looked up in table	8

Since raw statistics—such as source and destination IP addresses—can lead to data leakage and bias, they cannot be directly used for training classifiers. Therefore, we derived 11 informative features from the collected data for classifier training and

testing. The names and derivation formulas for these features are provided below:

1. *Num_of_Source_IP*:

$$\text{Num_of_Source_IP} = \sum_{i=1}^n \delta(\text{Source_IP}_i) \quad (3.1)$$

where:

- Source_IP_i represents the i -th source IP address in the collection.
- $\delta(\text{Source_IP}_i)$ is a function that counts each occurrence of a source IP address in the collection.
- n is the total number of flow entries in the flow table collected.

This is the total number of unique source IPs in every collection

2. *Num_of_Port*:

$$\text{Num_of_Source_Port} = \sum_{i=1}^n \delta(\text{Source_Ports}_i) \quad (3.2)$$

where:

- Source_Ports_i represents the i -th source port in the collection.
- $\delta(\text{Source_Ports}_i)$ is a function that counts each occurrence of a source port in the collection.
- n is the total number of flow entries in the flow table collected.

This is the total number of unique source ports in every collection.

3. *Paired IP to Num of Flow*:

$$\text{Ratio_of_Paired_Flow} = \frac{\Delta\text{Flow_Pair_Count}}{\Delta\text{Flow_Count}} \quad (3.3)$$

where:

- $\Delta\text{Flow_Pair_Count}$ represents the difference in the number of flow pairs (IPs recorded as both source and destination) between the current collection and the previous collection:

$$\Delta\text{Flow_Pair_Count} = \text{Flow_Pair_Count}_{\text{current}} - \text{Flow_Pair_Count}_{\text{previous}}$$

- $\Delta\text{Flow_Count}$ represents the difference in the total number of flows between the current collection and the previous collection:

$$\Delta\text{Flow_Count} = \text{Flow_Count}_{\text{current}} - \text{Flow_Count}_{\text{previous}}$$

This metric calculates the ratio of the change in the number of paired flows to the change in the total number of flows over consecutive collections. It provides insight into how the paired flow count evolves relative to the overall flow count.

4. *Change in Number of Incoming Packets:*

$$\text{Change.in.Num.of.Incoming.Packets} = \Delta\text{Total.Packets} \quad (3.4)$$

where:

- $\Delta\text{Total.Packets}$ represents the difference in the total number of packets across all flows between the current collection and the previous collection:

$$\Delta\text{Total.Packets} = \text{Total.Packets}_{\text{current}} - \text{Total.Packets}_{\text{previous}}$$

- Total.Packets is calculated as the sum of packets across all flows in a collection:

$$\text{Total.Packets} = \sum_{i=1}^n \text{Packets}_i$$

where:

- Packets_i is the total number of packets in the i -th flow.
- n is the total number of flows in the collection.

This feature measures the change in packet activity by computing the difference in the total number of packets observed between consecutive collections. The total number of packets for a collection is derived by summing the number of packets in each individual flow within that collection.

5. *Change in Number of Lookup:*

$$\text{Change.in.Num.of.Lookup} = \Delta\text{Lookup.Count} \quad (3.5)$$

where:

- $\Delta\text{Lookup_Count}$ represents the change in the number of lookup counts between the current collection and the previous collection:

$$\Delta\text{Lookup_Count} = \text{Lookup_Count}_{\text{current}} - \text{Lookup_Count}_{\text{previous}}$$

- $\text{Lookup_Count}_{\text{current}}$: The total lookup counts in the current collection.
- $\text{Lookup_Count}_{\text{previous}}$: The total lookup counts in the previous collection.

This feature measures the difference in the number of lookup operations performed between two consecutive collections. It provides insights into changes in lookup activity, which could indicate shifts in network traffic behavior or table usage patterns.

6. *Average Packets per Flow:*

$$\text{Average_Packets_per_Flow} = \frac{\text{Total_Packets}}{\text{Total_Flows}} \quad (3.6)$$

where:

- Total_Packets is the total number of packets in all flows for the collection:

$$\text{Total_Packets} = \sum_{i=1}^n \text{Packets}_i$$

where:

- Packets_i is the number of packets in the i -th flow.
- n is the total number of flows in the collection.
- Total_Flows is the total number of flows in the collection.

This feature calculates the average number of packets sent per flow in the collection, providing an insight into the flow's traffic intensity.

7. *Average Bytes per Flow:*

$$\text{Average_Bytes_per_Flow} = \frac{\text{Total_Bytes}}{\text{Total_Flows}} \quad (3.7)$$

where:

- Total_Bytes is the total number of bytes in all flows for the collection:

$$\text{Total_Bytes} = \sum_{i=1}^n \text{Bytes}_i$$

where:

- Bytes_{*i*} is the number of bytes in the *i*-th flow.
- *n* is the total number of flows in the collection.
- Total_Flows is the total number of flows in the collection.

This feature calculates the average number of bytes transmitted per flow in the collection, providing insight into the data volume handled by each flow.

8. *Standard Deviation of Packets:*

$$\text{Std_Dev_of_Packets} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\text{Packets}_i - \mu)^2} \quad (3.8)$$

where:

- *n* is the total number of flows in the collection.
- Packets_{*i*} is the number of packets in the *i*-th flow.
- μ is the mean number of packets per flow, calculated as:

$$\mu = \frac{1}{n} \sum_{i=1}^n \text{Packets}_i$$

This feature computes the dispersion or variability in the number of packets across all flows in the collection, providing insight into traffic distribution consistency.

9. *Standard Deviation of Bytes:*

$$\text{Std_Dev_of_Bytes} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\text{Bytes}_i - \mu)^2} \quad (3.9)$$

where:

- *n* is the total number of flows in the collection.
- Bytes_{*i*} is the number of bytes in the *i*-th flow.
- μ is the mean number of bytes per flow, calculated as:

$$\mu = \frac{1}{n} \sum_{i=1}^n \text{Bytes}_i$$

This feature computes the dispersion or variability in the number of bytes across all flows in the collection, providing insight into how data volume is distributed among the flows.

10. *Average Duration per Flow:*

$$\text{Average_Duration_per_Flow} = \frac{\text{Total_Duration}}{\text{Total_Flows}} \quad (3.10)$$

where:

- Total_Duration is the sum of the duration of all flows in the collection:

$$\text{Total_Duration} = \sum_{i=1}^n \text{Duration}_i$$

where:

- Duration_{*i*} is the duration of the *i*-th flow.
- *n* is the total number of flows in the collection.
- Total_Flows is the total number of flows in the collection.

This feature calculates the average time that each flow lasts within the collection, providing insight into the flow duration characteristics.

11. *Protocol:* This feature represents the protocol of collected traffic. In this work, there are 3 possible values of it, 1 for ICMP, 6 for TCP and 17 for UDP.

Algorithm 3 outlines the feature extraction process. The Ryu controller sends OFPFlowStatsRequest and OFPTableStatsRequest messages to the data plane, prompting switches to reply with OFPFlowStatsReply and OFPTableStatsReply messages that contain the raw traffic statistics. The OFPFlowStatsReply includes flow-related details like IP addresses, port numbers, and packet counts, while the OFPTableStatsReply provides statistics on flow table lookups. These statistics are then processed to generate the 11 derived features, which are saved in a CSV file for subsequent training and testing.

Algorithm 3: Feature Extraction for Network Traffic

Input: Traffic in the network
Output: Dataset with 11 extracted features

```

1  foreach datapath  $p$  in {datapath_list} do
2      OFPFlowStatsRequest(datapath) ;                      // Request flow stats from switches
3      OFPTableStatsRequest(datapath) ;                      // Request table stats from switches
4      lookup_count  $\leftarrow$  0
5      src_ip_list, dst_ip_list, src_port_list  $\leftarrow$  {}
6      previous_stats  $\leftarrow$  {"packet_count": 0, "lookup_count": 0, "pair_count": 0, "flow_count": 0}
7      packet_count, byte_count, src_port, ip_proto, duration  $\leftarrow$  0
8      flow_packets, flow_bytes  $\leftarrow$  []
9      lookup_count  $\leftarrow$  TableStatsReplyHandler.lookup_count
10     foreach stat in en.msg.body do
11         packet_count += stat.packet_count
12         byte_count += stat.byte_count
13         duration += stat.duration_sec + stat.duration_nsec / 1e9
14         ip_proto  $\leftarrow$  stat.match['ip_proto']
15         src_ip_list.add(stat.match['ipv4_src'])
16         dst_ip_list.add(stat.match['ipv4_dst'])
17         src_port_list.add(stat.match['ports'])
18         flow_packets.append(stat.packet_count)
19         flow_bytes.append(stat.byte_count)

        // Compute derived features
20     average_packet_count  $\leftarrow$  packet_count / flow_count
21     average_byte_count  $\leftarrow$  byte_count / flow_count
22     packet_count_diff  $\leftarrow$  |packet_count - previous_stats["packet_count"]|
23     lookup_count_diff  $\leftarrow$  |lookup_count - previous_stats["lookup_count"]|
24     pair_count  $\leftarrow$  src_ip_list  $\times$  dst_ip_list
25     pair_count_diff  $\leftarrow$  |pair_count - previous_stats["pair_count"]|
26     flow_count_diff  $\leftarrow$  |flow_count - previous_stats["flow_count"]|
27     if flow_count > 0 then
28         dur_per_flow  $\leftarrow$  duration / flow_count
29     else
30         dur_per_flow  $\leftarrow$  0
31     if flow_count_diff > 0 then
32         pair_count_ratio  $\leftarrow$  pair_count_diff / flow_count_diff
33     else
34         pair_count_ratio  $\leftarrow$  0
35     if flow_packets not empty then
36         packet_std_dev  $\leftarrow$  std(flow_packets)
37     else
38         packet_std_dev  $\leftarrow$  0
39     if flow_bytes not empty then
40         byte_std_dev  $\leftarrow$  std(flow_bytes)
41     else
42         byte_std_dev  $\leftarrow$  0
43     WriteInFile(src_ip_count, src_port_count, pair_count_ratio, packet_count_diff,
44                 lookup_count_diff, ip_proto, average_packet_count, average_byte_count,
45                 packet_std_dev, byte_std_dev, dur_per_flow)

```

3.4 Feature analysis

To analyze feature behavior under both normal and attack conditions, a small dataset was generated using the process outlined in Figure 3.3. This dataset includes 10 selected features and 90 records, excluding the ‘Protocol’ feature since its value remains constant regardless of traffic conditions. The dataset comprises 30 records of normal traffic, followed by 30 records of attack traffic, and concludes with an additional 30 records of normal traffic.

Figure 3.4 illustrates how each feature exhibits distinct changes during an attack, indicating their usefulness in identifying malicious traffic. For example, the ‘Count_of_Source_IP’ feature does not immediately return to normal levels after an attack because the corresponding attack flows persist in the flow table until they expire after the idle timeout period.

3.5 Dataset preprocessing

Since all the features in generated dataset are numeric, there is no need to encode any of them. We noticed that there is huge gap between the values of features, MinMax normalization is adapted to preprocess the dataset. Formula of MinMax normalization is provided below.

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (3.11)$$

3.6 Evaluation of classifiers

Two datasets were created for training and testing the classifiers—one for binary classification and another for multi-class classification. Both datasets include 12 features (11 derived features plus a label). The binary dataset contains 140,000 records, evenly split between normal and attack traffic. The multi-class dataset comprises 60,000 records, with 10,000 normal records and 10,000 records for each attack type (SYN flood, UDP flood, ICMP flood, botnet-based, and FIN flood). Each dataset was divided into 80% training and 20% unseen testing sets, and a 10-fold cross-validation was applied during model evaluation.

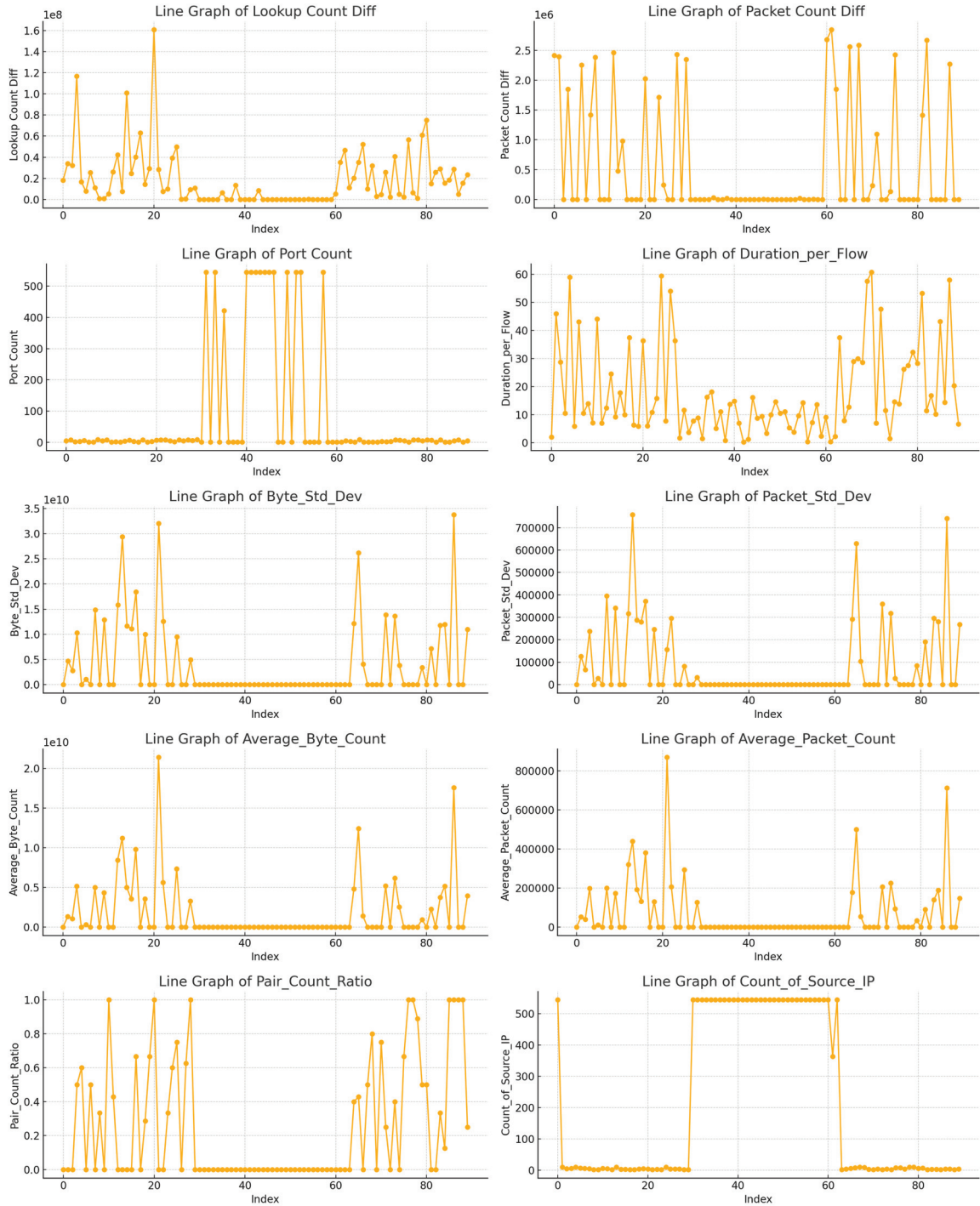


Figure 3.4: Behavior of features during normal and attack

Performance was measured using four metrics: accuracy, precision, recall, F1 score and training time. Classifier outcomes are categorized as follows:

- **True Positives (TP):** An attack traffic is predicted as an attack by the model.

- **True Negatives (TN):** A normal traffic is predicted as normal by the model.
- **False Positives (FP):** A normal traffic is incorrectly predicted as attack. This is referred to as a Type I error.
- **False Negatives (FN):** An attack traffic is incorrectly predicted as normal. This is referred to as a Type II error.

The four metrics are calculated based on these four conditions. The explanations and equations of them are provided below.

- **Accuracy:** This measures the overall correctness of a classifier by calculating the ratio of correct predictions to the total number of cases. This metric provides a general measure of performance; however, it may not be sufficient in cases of imbalanced datasets. It is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.12)$$

- **Precision:** Precision, also known as the positive predictive value, quantifies the accuracy of positive predictions. High precision indicates a low proportion of false positive predictions among the positive predictions made by the model. It is given by:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3.13)$$

- **Recall:** Recall, also referred to as sensitivity or the true positive rate, measures the ability of a classifier to identify all relevant positive cases. A high recall indicates that the model successfully captures most of the actual positive instances. It is defined as:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3.14)$$

- **F1 score:** The F1 score is the harmonic mean of precision and recall, providing a single metric that balances both aspects. This measure is particularly useful when the class distribution is imbalanced, as it accounts for both false positives and false negatives. It is computed as:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.15)$$

Table 3.3 and Table 3.4 compare the performance of 9 classifiers in binary classification and multi-class classification using four metrics. In binary classification, Random Forest achieves the top accuracy (98.74%) and F1 (98.76%) in 136 seconds of training, while XGB matches closely (98.69% accuracy, 98.71% F1) in only 3 seconds, and LightGBM likewise combines high accuracy (98.65%) with a brief 11 seconds training time. These methods also exhibited high recall values, indicating robust capability in correctly identifying positive instances. By contrast, Naive Bayes, although the fastest to train (0.6 seconds), exhibits the poorest predictive metrics (93.11% accuracy, 93.55% F1), highlighting its limited suitability for this dataset. The remaining classifiers (SVM, ANNs, KNN, Logistic Regression, and Decision Tree) displayed competitive performance, with accuracies ranging between approximately 97.55% and 98.39%. Support Vector Machines and the Artificial Neural Network also exceed 98% accuracy, but incur very long training times (3 838 seconds and 4 391 seconds, respectively), making them less efficient for large-scale deployment.

In the multi-class setting, gradient-boosting algorithms deliver the best trade-off between accuracy and efficiency: XGB achieves 95.51% accuracy and a 95.39% F1 score in just 5 seconds of training, while LightGBM follows closely with 95.48% accuracy and a 95.35% F1 score in 9 seconds. Random Forest remains a robust alternative, attaining 94.36% accuracy and 94.23% F1 in 75 seconds. In contrast, SVM and ANNs exceed 92% accuracy (92.64% and 93.34%, respectively) but incur substantial computational costs (431 seconds and 2 129 seconds), which may limit their practicality. Simpler models such as Decision Tree, KNN, Logistic Regression, and Naive Bayes train in under 32 seconds but achieve lower accuracy (89.85%–92.30%) and F1 scores (88.70%–92.24%), underscoring the inherent trade-off between model complexity, predictive performance, and training time.

3.7 Strategies to optimize performance

3.7.1 Feature selection

To enhance model performance in both binary and multi-class classification, feature selection is applied. Since not all features contribute equally to predictive power and irrelevant features may introduce noise, focusing on the most informative aspects of

Table 3.3: Binary classification performance of nine classifiers.

Classifier	Accuracy	Precision	Recall	F1 Score	Training Time (seconds)
Decision Tree	97.85%	97.84%	97.91%	97.88%	4
Random Forest	98.74%	97.95%	99.58%	98.76%	136
LightGBM	98.65%	97.80%	99.55%	98.67%	11
SVM	98.08%	97.60%	98.62%	98.11%	3838
ANNs	98.39%	97.61%	99.23%	98.41%	4391
Naive Bayes	93.11%	88.73%	98.92%	93.55%	0.6
Logistic Regression	97.55%	97.62%	97.53%	97.58%	2
KNN	98.28%	97.71%	98.90%	98.30%	177
XGB	98.69%	97.85%	99.58%	98.71%	3

Table 3.4: Multi-class classification performance of nine classifiers.

Classifier	Accuracy	Precision	Recall	F1 Score	Training Time (seconds)
Decision Tree	92.24%	92.24%	92.24%	92.24%	2
Random Forest	94.36%	94.45%	94.36%	94.23%	75
LightGBM	95.48%	95.65%	95.48%	95.35%	9
SVM	92.64%	93.24%	92.64%	92.36%	431
ANNs	93.34%	94.00%	93.34%	93.11%	2129
Naive Bayes	89.85%	91.01%	89.85%	88.70%	0.5
Logistic Regression	91.93%	92.37%	91.93%	91.62%	4
KNN	92.30%	92.26%	92.30%	92.21%	32
XGB	95.51%	95.70%	95.51%	95.39%	5

the data is essential to reduce overfitting and improve generalization.

First, Weighted Feature Importance (WFI) scores are obtained for the top-performing classifiers—Random Forest (RF) and XGBoost (XGB)—in both classification settings. These scores, presented in Tables 3.5 and 3.6, indicate each feature’s contribution to the model’s prediction.

Table 3.5: WFI scores for Random Forest (binary classification).

Feature	Importance
Count of Source IP	0.283648
Average Packet Count	0.282949
Packet Std Dev	0.116003
Duration per Flow	0.069137
Lookup Count Diff	0.068215
Packet Count Diff	0.066524
Byte Std Dev	0.044001
Average Byte Count	0.029721
Pair Count Ratio	0.022178
Port Count	0.011120
Protocol	0.006504

Recursive Feature Elimination (RFE) is a method of feature selection, it iteratively train the model, eliminate the lowest ranking features until the optimal subset of features is found [24]. In this work, Recursive Feature Elimination is used to iteratively remove the feature with the lowest WFI score, while monitoring model performance with accuracy and F1 score. As illustrated in Figure 3.5, both RF and XGB perform best when all 11 features are retained, indicating that each feature is informative enough.

3.7.2 Weighted Majority Vote Ensemble method

Ensemble learning involves merging several individual classifiers and determining the final class label by taking a majority vote. However, since not all classifiers perform

Table 3.6: WFI scores for XGB (multi-class classification).

Feature	Importance
Duration per Flow	0.199222
Lookup Count Diff	0.156889
Packet Count Diff	0.156000
Average Packet Count	0.098167
Count of Source IP	0.088667
Port Count	0.065167
Packet Std Dev	0.060444
Average Byte Count	0.060389
Pair Count Ratio	0.047333
Byte Std Dev	0.038889
Protocol	0.028833

equally well, it’s advantageous to assign them different weights to boost overall classification accuracy [15]. In this work, a Genetic Algorithm (GA) [40] is employed to optimize the weights assigned to individual classifiers in a weighted majority vote ensemble. The implementation leverages the DEAP library to define a fitness function that measures the ensemble’s accuracy on a validation dataset. Each candidate solution is represented as a chromosome—a vector of weights corresponding to the classifiers—and an initial population of these weight vectors is randomly generated. The GA uses several key genetic operators to explore the search space:

- **Crossover (Mate):** A blend crossover operator (cxBlend) is applied to combine portions of two parent individuals, allowing offspring to inherit characteristics from both parents.
- **Mutation:** Gaussian mutation (mutGaussian) introduces small random perturbations to individual weights, maintaining genetic diversity and helping to avoid local optima.
- **Selection:** Tournament selection (selTournament) is used to choose the fittest individuals based on the accuracy evaluation, ensuring that better-performing

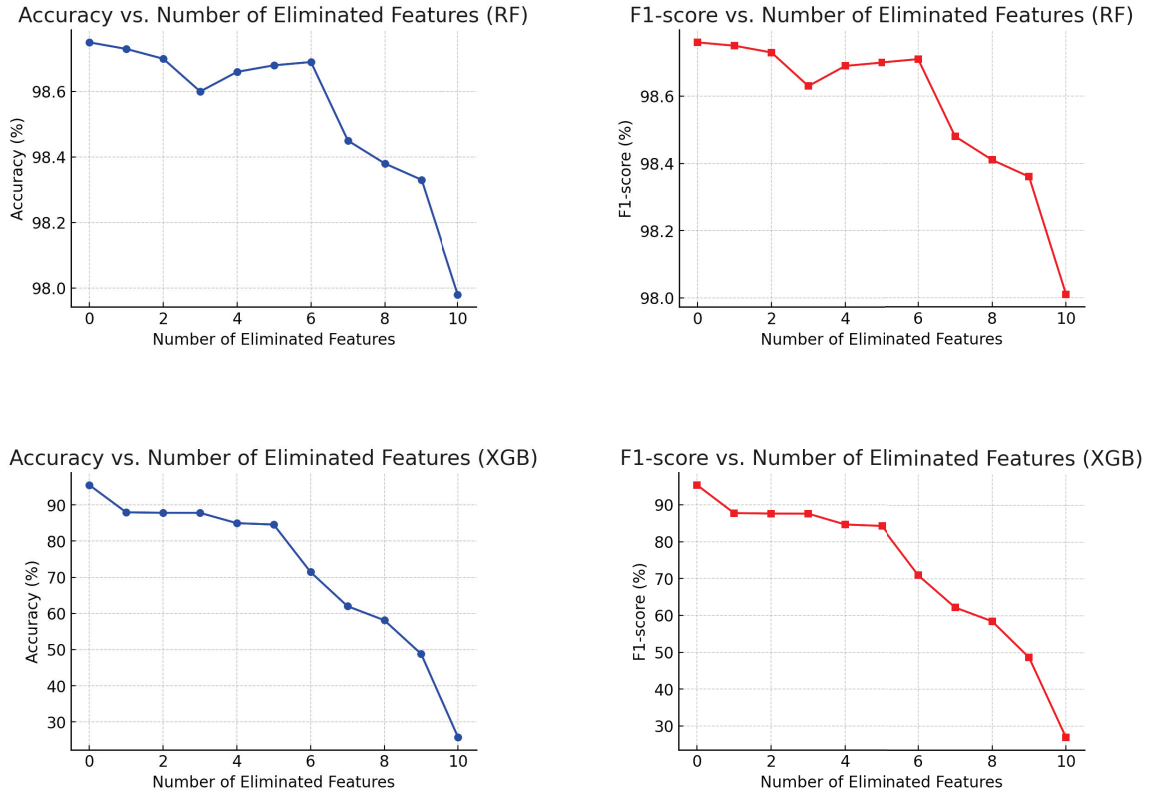


Figure 3.5: Evaluation of feature subsets using RFE

weight vectors have a higher chance of propagating to subsequent generations.

Over successive generations—each defined by iterative application of these operators—the GA progressively refines the population. The process ultimately converges on an optimal or near-optimal set of weights that, when normalized to sum to one, maximizes the ensemble’s predictive performance. The optimal weight sets of binary and multi-class prediction are shown in Table 3.7.

After applying weighed majority vote ensemble method, there was an improvement in the performance for both binary and multi-class classification in terms of all metrics in this work. The result is shown in Table 3.8. The majority vote classifier, optimized via a genetic algorithm, exhibits a consistent performance advantage over individual ensemble methods in both binary and multi-class classification tasks. Specifically, In the binary task, the majority vote ensemble attains 98.78% accuracy and a 98.79 % F1 score—improvements of 0.04% and 0.03%, respectively, compared

Table 3.7: Optimized Classifier Weights for Binary and Multi-class Classification

Classifier	Binary Weight	Multi-class Weight
ANNs	0.04337	-0.08076
KNN	0.00560	-0.20395
XGB	0.47038	0.48565
DT	0.43642	-0.00648
NB	-0.35001	0.04383
LGB	0.03878	0.52982
LR	-0.14712	0.15286
RF	0.60582	0.04341
SVM	-0.10324	0.03563

with Random Forest—alongside a precision increase to 97.99 % (+0.04%) and recall to 99.60% (+0.02%). These gains require 245 seconds of training, versus 136 seconds for Random Forest. In the multi-class scenario, the majority vote approach achieves higher accuracy (95.57% vs. 95.51%) and F1 score (95.45% vs. 95.39%) than XGB, reflecting its robust capability to handle more complex label distributions at a training time of 156 seconds compared with XGB’s 5 seconds. Although the performance gains are modest, they underscore the efficacy of combining multiple classifiers through majority voting, particularly when further optimized by genetic algorithms. This ensemble-based strategy leverages the diversity of base models, leading to improvements in predictive power over any single best-performing classifier, albeit with increased computational overhead.

Table 3.8: Performance metrics for majority vote ensemble learning in binary and multi-class classification.

Classifier	Accuracy	Precision	Recall	F1 Score	Training Time (seconds)
Majority vote (Binary)	98.78%	97.99%	99.60%	98.79%	245
Random Forest (Binary)	98.74%	97.95%	99.58%	98.76%	136
Majority vote (Multi-class)	95.57%	95.77%	95.57%	95.45%	156
XGB (Multi-class)	95.51%	95.70%	95.51%	95.39%	5

Chapter 4

DDoS Mitigation

This chapter details how micro-segmentation and Attribute-Based Access Control (ABAC) are implemented in SDNs to mitigate DDoS attacks. First, the deployment of micro-segmentation according to the proposed methodology is described. Next, the use of ABAC to manage communication within and between micro-segments is explained. A use case is then presented to demonstrate the benefits of this mitigation strategy.

Figure 4.1 illustrates the architecture of the proposed micro-segmentation. Several functions have been integrated into the Ryu controller to enhance SDN security, with proposed ML-based IDS already discussed in Chapter 3. ABAC automates network management by interacting with the ML-based IDS function. For example, when an attacker is detected by IDS in controller, controller will configure ABAC policies and apply them to corresponding micro-segments to mitigate the effect of the attacker. The micro-segmentation topology and ABAC attributes are predefined and implemented within the controller.

4.1 Micro-segmentation

Micro-segmentation is a fundamental strategy in DDoS mitigation that involves dividing the network into smaller, isolated segments. This approach limits the lateral movement of an attacker and minimizes the impact of a potential breach. The life cycle of micro-segmentation is shown in Figure 4.2.

The first step to deploy micro-segmentation to a network is resource and requirements identification. This step involves systematically identifying all network resources, such as servers, databases, applications and endpoints. Each assets should be evaluated based on its criticality and sensitivity to security events and daily use. This would provide a comprehensive inventory that forms the basis for targeted security control. All the following processes should stick to this inventory to have a

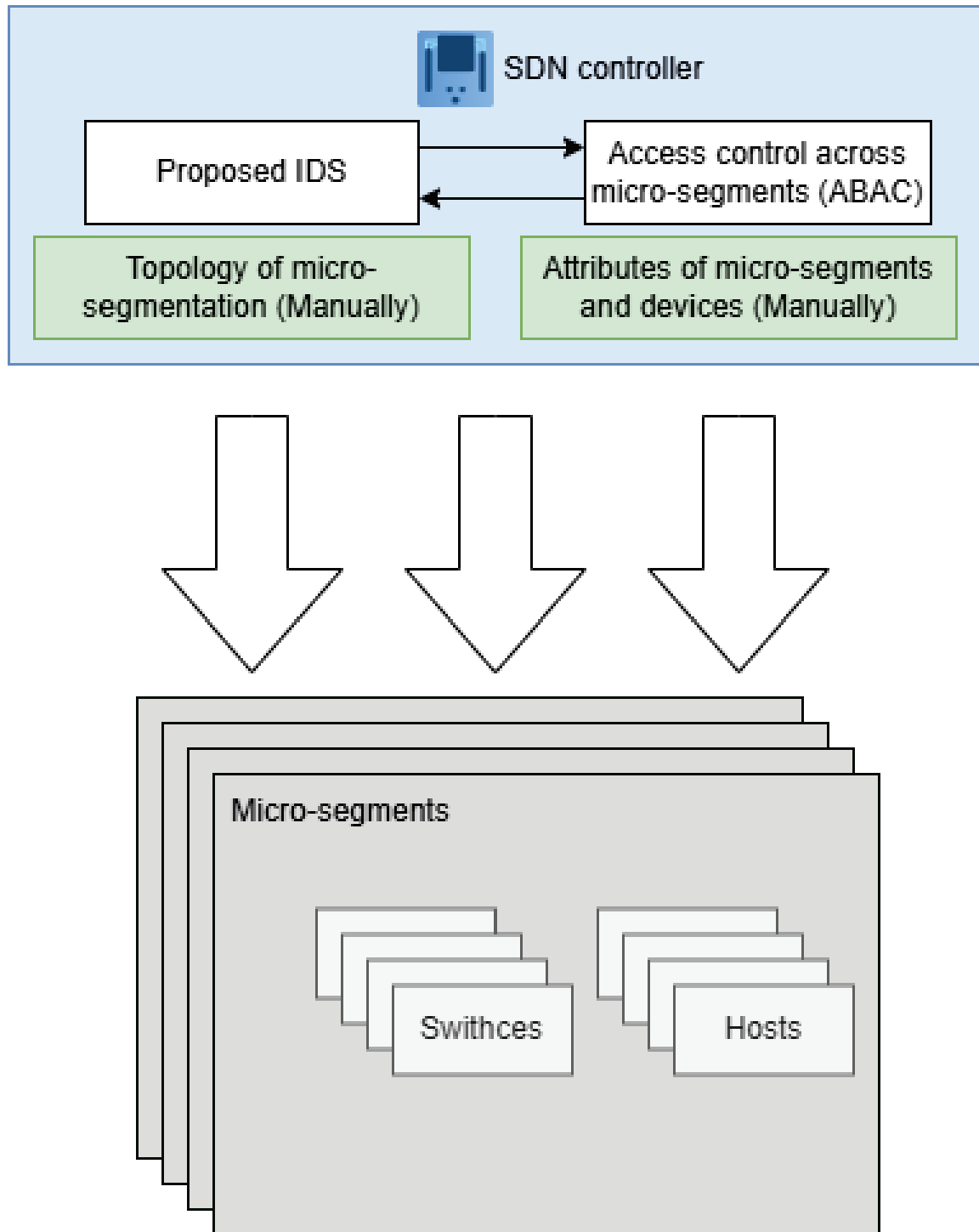


Figure 4.1: Architecture of proposed micro-segmentation

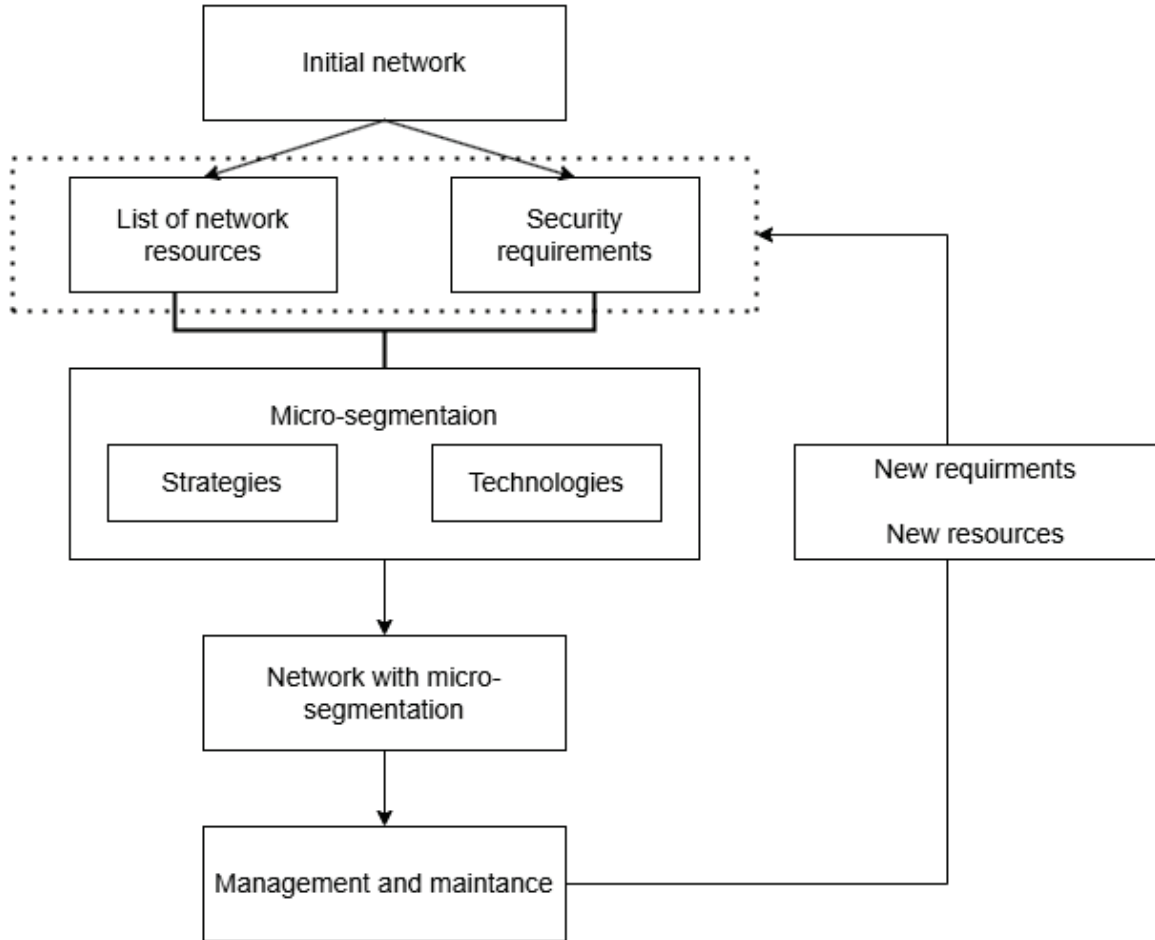


Figure 4.2: Life cycle of micro-segmentation

satisfied result.

After that, some strategies and technologies of micro-segmentation should be adopted. In this phase, how the resources should be grouped in network micro-segments should be determined. And the security strategies for each micro-segments should be clearly defined, followed by customized security polices. The policies may includes acceptable traffic, access controls and monitoring requirements. The technologies to do so need to be selected in this phase as well. People should decide what technologies should be adopted in order to secure and manage the network. For example, SDN is utilized to realize the micro-segmentation and ABAC for the access control in this work. Also, technology selection really depends on the strategies.

Micro-segmentation starts by answering how the network resources should be grouped. A great answer here would be a topology of micro-segmentation which shows

access control between protected resources. Topology design for micro-segmentation can be either automated or manual. Machine learning models or mathematical algorithms are usually applied to automate micro-segmentation topology. Automated topology is a good choice when the scale of network is large. Because when there are more resources and requirements, it is more complicated to find optimal topology. And the network is more vulnerable when there is an error in configuration. However, manual topology design can be a better option for medium or small network since there is no worry of complexity.

After establishing the topology, the next step is choosing the deployment technology. Options include VLANs, cloud-based segmentation tools, and SDN. SDN offers significant advantages due to its manageability, scalability, and simplified configuration for both topology deployment and access control management. Especially in IoT environment, in addition to NFV, software defined attributes of SDN can also be useful for micro-segmentation [59].

After all, the network should be secured by micro-segmentation. But this does not bring the end of micro-segmentation. Micro-segmentation is an ongoing process that requires continuous management and maintenance. As new resources join the network or as security requirements evolve, the micro-segmentation framework must be updated to ensure sustained protection. The whole process of micro-segmentation can be also seen as a function which takes origin network as input and segmented network as output. Any changes to the input may have effects on the output.

4.2 Attribute Based Access Control (ABAC)

Since micro-segmentation is deployed in SDN, controller needs to manage micro-segments by controlling the traffic between them. Prerequisites of access control should be established when topology of micro-segmentation is designed. There are different access control technologies based on the technologies of topology deployment. For example, you can use firewalls in VLANs if the environment supports it. Or DPI engines can be used in cloud-based platform. In this work, SDN supports various access control technologies such as ACLs, dynamic firewalls, Role Based Access Control, ABAC and so on. ABAC is adopted in this work in consideration of dynamic access control decisions are required. And by using well-defined attributes

that apply to both subjects and objects, authentication and authorization processes can be carried out and managed within the same infrastructure or across different systems, all while ensuring adequate security levels are upheld.

In this work, Casbin is used to implement ABAC in Ryu controller. Casbin is an open-source, lightweight, and highly flexible access control library that supports multiple authorization models including Access Control Lists (ACL), Role-Based Access Control, and Attribute-Based Access Control. Originally written in Go and now available for multiple programming languages, it lets developers define and enforce security policies via customizable model and policy files, making it easy to integrate into diverse applications and distributed systems.

For ABAC, Casbin leverages what's known as its PERM model—a framework that evaluates access decisions based on dynamic attributes of the subject (user), object (resource), and the requested action [37]. In this model, instead of relying solely on static roles or permissions, you can define policy rules that incorporate various attributes and conditions. At runtime, when a request is made, Casbin's enforcer evaluates these attributes using built-in or custom functions, allowing for fine-grained, context-sensitive access control decisions that adapt to complex business logic and changing environments. More details of PERM model can be accessed in [9].

The PERM model has four foundations: Policy, Effect, Request and Matcher [9]:

- **Request:** It defines the request parameters. A basic request is a tuple object, requiring at least a subject (accessed entity), object (accessed resource), and action (access method).
- **Policy:** It defines the model for the access strategy. It specifies the name and order of the fields in the Policy rule document.
- **Matcher:** It defines the matching rules for Request and Policy.
- **Effect:** It performs a logical combination judgment on the matching results of Matchers.

4.3 Use Case

This section illustrates the proposed DDoS mitigation method using a practical example. In this scenario, a company operates an SD-IoT network for internal purposes, with no external access. The company plans to implement the proposed micro-segmentation strategy to secure their network.

The network inventory is created by identifying key resources and their requirements:

- A major database contains the data log of industry, which is very sensitive. Only chief operator can access to the data in it.
- IoT devices, such as sensors, cameras, can upload data to the major database.
- A database and a server are for HR. Only two supervisors can access to them.

According to the inventory for micro-segmentation, a topology has been designed for the network. The network is illustrated in Figure 4.3 after topology realization.

Access controls between the micro-segments are managed using Attribute-Based Access Control. The Ryu controller oversees the network and enforces policies. The resources are grouped as follows:

- All IoT devices are grouped in micro-segment 1
- The major database in micro-segment 2, the chief operator in micro-segment 3
- HR database in micro-segment 4, HR server in micro-segment 5, two supervisor are put in micro-segment 6 and 7 respectively.

ABAC policies, based on the inventory, govern access between these segments:

- Micro-segment 1 (IoT) and micro-segment 2 (major database).
- Micro-segment 3 (chief operator) and micro-segment 2 (major database).
- Micro-segment 4 (HR database) and both micro-segment 6, 7 (supervisors).
- Micro-segment 5 (HR server) and both micro-segment 6, 7 (supervisors).

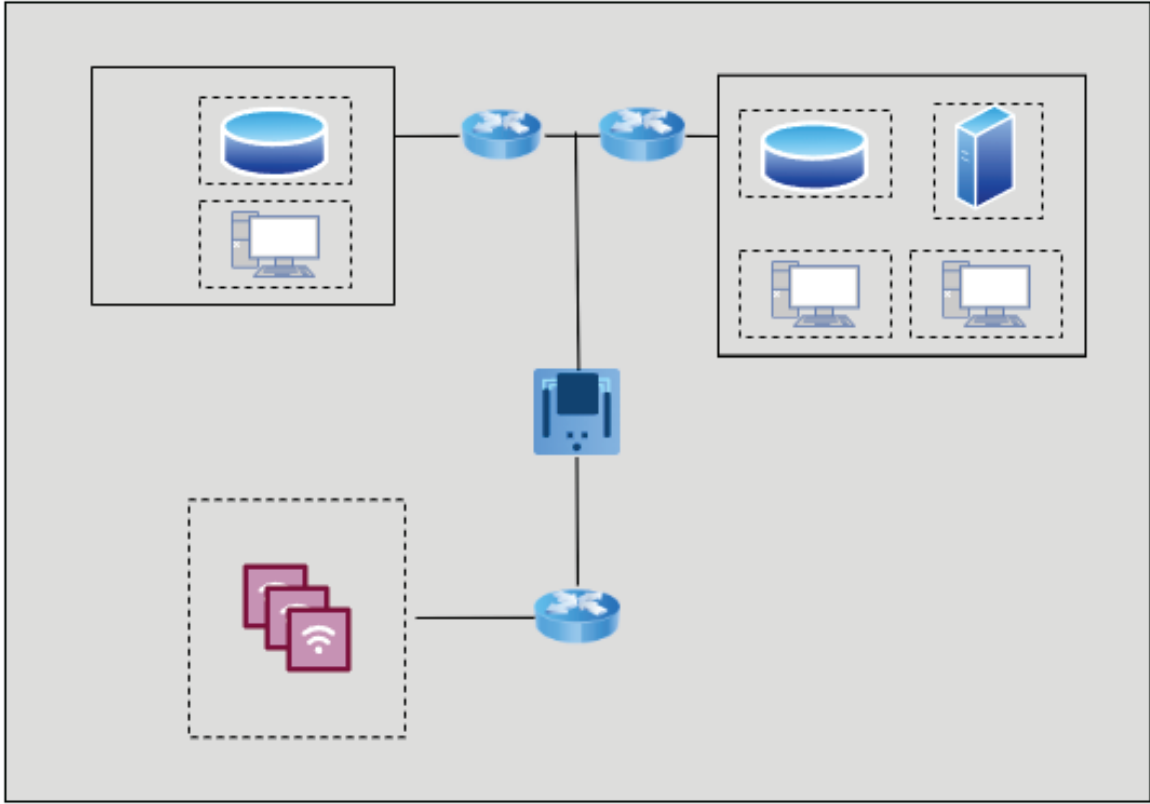


Figure 4.3: Example of micro-segmentation

Given that IoT devices are resource-constrained and less capable of implementing security mechanisms, they are particularly vulnerable to attacks. If an attacker compromises IoT devices, turning them into botnets to launch a DDoS attack against the major database, it could result in significant damage to the company. ABAC, configured within the controller, helps prevent such incidents. The configuration of ABAC is shown in Table 4.1.

Table 4.1: Configuration of ABAC

Name	Configuration
request_definition	<code>r = sub, obj, act</code>
policy_definition	<code>p = attack_ip, attack_segment, act, eft</code>
policy_effect	<code>e = !some(where (p.eft == deny))</code>
matchers	<code>m = r.sub.ip == p.attack_ip r.sub.micro_segment == p.attack_segment r.obj.ip == p.attack_ip r.obj.micro_segment == p.attack_segment && r.act == p.act</code>

In this configuration, two attributes—IP address and micro-segment—are defined. For each access request, the controller checks the attributes of both the subject and object. If no matching policies are found with an effect of “deny”, the request is permitted.

If an IoT device is compromised and used to launch an attack on the major database, the controller detects the attack immediately, updates the ABAC policies with the attacker’s IP address and micro-segment, and blocks the compromised micro-segment. Additionally, other IoT devices in the same segment may also be compromised, resulting in the entire segment being blocked to prevent further damage.

This approach offers two significant benefits in this case: it mitigates the impact of DDoS attacks through effective micro-segmentation combined with ABAC, and it establishes a defense-in-depth strategy that enhances overall network resilience.

Chapter 5

Conclusion

This thesis makes significant contributions to both the detection and mitigation of DDoS attacks in SD-IoT networks.

Proposed algorithms were used to generate traffic and compile two datasets in CSV format within a simulated SDN. The generated traffic spans multiple protocols, including ICMP, UDP, and TCP, and covers various attack types such as SYN flood, UDP flood, ICMP flood, botnet-based, and FIN flood. From the collected data, 17 raw traffic statistics were extracted and processed to derive 11 informative features, with analysis confirming their strong indication of DDoS activity.

Two datasets were created for binary and multi-class classification, respectively. The preprocessed data was then used to train various machine learning classifiers, including Decision Tree, LightGBM, XGBoost, Random Forest, SVM, KNN, Naive Bayes, Artificial Neural Network, and Logistic Regression. Evaluation using accuracy, precision, recall, and F1 score revealed that Random Forest and XGBoost achieved the highest performance, with Random Forest reaching 98.74% accuracy for binary classification and XGBoost attaining 95.51% accuracy for multi-class classification.

To further enhance detection performance, feature selection was applied to reduce noise. Recursive Feature Elimination based on Weighted Feature Importance confirmed that all 11 features were essential. In addition, a weighted majority vote ensemble method was implemented, with classifier weights optimized using genetic algorithms applied to the best-performing models (RF and XGB). This ensemble approach led to improvements across all evaluation metrics.

Finally, a DDoS mitigation methodology is proposed that integrates micro-segmentation with Attribute-Based Access Control in SDNs. The deployment strategy for micro-segmentation is detailed, and the Casbin tool is introduced to implement ABAC. A use case demonstrates that this approach not only mitigates the impact of DDoS attacks but also establishes a robust defense-in-depth framework.

While this thesis provides valuable contributions to dataset generation, DDoS detection, and mitigation, several limitations remain that could be addressed in future work.

In SDN architecture, a single point of failure can occur if the controller is attacked. To prevent this, it is crucial to monitor the communication between switches and the controller. The use of distributed controllers offers a promising solution to mitigate this risk, as they can provide redundancy and resilience.

Leveraging distributed controllers in SDN can also enhance attack mitigation strategies. Each controller can be assigned to monitor and manage its own micro-segments, allowing DDoS attacks to be addressed more efficiently. For instance, certain micro-segments may occasionally experience high traffic volumes, and the IDS deployed on that controller could quickly identify DDoS attacks based on the specific characteristics of that traffic.

As SDN scales, manually designing micro-segmentation topologies becomes increasingly inefficient. Machine learning algorithms with high performance can be integrated into the controller to optimize the micro-segmentation topology automatically. This approach not only improves efficiency but also reduces the potential for human error, which can have significant consequences in large-scale networks.

In conclusion, this thesis introduces a comprehensive framework for detecting and mitigating DDoS attacks. The proposed feature extraction and dataset generation methods offer practical applications, while the DDoS mitigation strategy demonstrates the potential of micro-segmentation and Attribute-Based Access Control (ABAC) in enhancing SD-IoT security.

Bibliography

- [1] Zakaria Abou El Houda, Lyes Khoukhi, and Abdelhakim Senhaji Hafid. Bringing intelligence to software defined networks: Mitigating ddos attacks. *IEEE Transactions on Network and Service Management*, 17(4):2523–2535, 2020.
- [2] Nisha Ahuja, Gaurav Singal, Debajyoti Mukhopadhyay, and Neeraj Kumar. Automated ddos attack detection in software defined networking. *Journal of Network and Computer Applications*, 187:103108, 2021.
- [3] Amir Ali and Muhammad Murtaza Yousaf. Novel three-tier intrusion detection and prevention system in software defined network. *IEEE Access*, 8:109662–109676, 2020.
- [4] Shruti Banerjee and Partha Sarathi Chakraborty. To detect the distributed denial-of-service attacks in sdn using machine learning algorithms. In *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 966–971. IEEE, 2021.
- [5] Narmeen Zakaria Bawany, Jawwad A Shamsi, and Khaled Salah. Ddos attack detection and mitigation using sdn: methods, practices, and solutions. *Arabian Journal for Science and Engineering*, 42:425–441, 2017.
- [6] Kamal Benzekki, Abdeslam El Fergougui, and Abdelbaki Elbelrhiti Elalaoui. Software-defined networking (sdn): a survey. *Security and communication networks*, 9(18):5803–5833, 2016.
- [7] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.
- [8] Wolfgang Braun and Michael Menth. Software-defined networking using open-flow: Protocols, applications and architectural design choices. *Future Internet*, 6(2):302–336, 2014.
- [9] Casbin. How it works. <https://casbin.org/docs/how-it-works>, 2025. Accessed: 2025-03-24.
- [10] Zhuo Chen, Fu Jiang, Yijun Cheng, Xin Gu, Weirong Liu, and Jun Peng. Xgboost classifier for ddos attack detection and analysis in sdn-based cloud. In *2018 IEEE international conference on big data and smart computing (bigcomp)*, pages 251–256. IEEE, 2018.
- [11] Ryu SDN Framework Community. Ryu sdn framework. <https://github.com/faucetsdn/ryu>, 2025. Accessed: 2025-03-04.

- [12] Jie Cui, Mingjun Wang, Yonglong Luo, and Hong Zhong. Ddos detection and defense mechanism based on cognitive-inspired computing in sdn. *Future generation computer systems*, 97:275–283, 2019.
- [13] Barry De Ville. Decision trees. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(6):448–455, 2013.
- [14] V Deepa, K Muthamil Sudar, and P Deepalakshmi. Design of ensemble learning methods for ddos detection in sdn environment. In *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, pages 1–6. IEEE, 2019.
- [15] Alican Dogan and Derya Birant. A weighted majority voting ensemble approach for classification. In *2019 4th international conference on computer science and engineering (UBMK)*, pages 1–6. IEEE, 2019.
- [16] AD Dongare, RR Kharde, Amit D Kachare, et al. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, 2(1):189–194, 2012.
- [17] K. Ekambaram and M. Varun. Microsegmentation: Defense in depth. *Dell Technologies Proven Professional Knowledge Sharing*, pages 1–8, 2021. [Online]. Available: <https://education.dell.com/content/dam/dell-emc/documents/en-us/2021KS-Ekambaram-Microsegmentation-Defense-in-Depth.pdf>.
- [18] JFJ Fan, GYJ Fan, and JGG Yang. Ddos attack detection system based on rf-svm-il model under sdn. *J. Comput. Sci*, 32:031–043, 2021.
- [19] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to sdn: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, 44(2):87–98, 2014.
- [20] Deyun Gao, Zehui Liu, Ying Liu, Chuan Heng Foh, Ting Zhi, and Han-Chieh Chao. Defending against packet-in messages flooding attack under sdn context. *Soft Computing*, 22:6797–6809, 2018.
- [21] Xianwei Gao, Chun Shan, Changzhen Hu, Zequn Niu, and Zhen Liu. An adaptive ensemble machine learning model for intrusion detection. *Ieee Access*, 7:82512–82521, 2019.
- [22] Xianjun Geng and Andrew B Whinston. Defeating distributed denial of service attacks. *It Professional*, 2(4):36–42, 2000.
- [23] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM computer communication review*, 38(3):105–110, 2008.

- [24] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46:389–422, 2002.
- [25] Shahzeb Haider, Adnan Akhunzada, Ghufraan Ahmed, and Mohsin Raza. Deep learning based ensemble convolutional neural network solution for distributed denial of service detection in sdns. In *2019 UK/China Emerging Technologies (UCET)*, pages 1–4. IEEE, 2019.
- [26] Joseph M Hilbe. Logistic regression. *International encyclopedia of statistical science*, 1:15–32, 2011.
- [27] Chung-Hsing Hsu and Ulrich Kremer. Iperf: A framework for automatic construction of performance prediction models. In *Workshop on Profile and Feedback-Directed Compilation (PFDC), Paris, France*. Citeseer, 1998.
- [28] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162):1–54, 2013.
- [29] Vikramaditya Jakkula. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 37(2.5):3, 2006.
- [30] Joelle Kabdjou and Norihiko Shinomiya. Improving quality of service and https ddos detection in mec environment with a cyber deception-based architecture. *IEEE Access*, 2024.
- [31] Mohammed Ibrahim Kareem and Mahdi Nsaif Jasim. Ddos attack detection using lightweight partial decision tree algorithm. In *2022 International Conference on Computer Science and Software Engineering (CSASE)*, pages 362–367. IEEE, 2022.
- [32] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.
- [33] Walid I Khedr, Ameer E Gouda, and Ehab R Mohamed. Fmdadm: A multi-layer ddos attack detection and mitigation framework using machine learning for stateful sdn-based iot networks. *IEEE Access*, 11:28934–28954, 2023.
- [34] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.
- [35] Majd Latah and Levent Toker. A novel intelligent approach for detecting dos flooding attacks in software-defined networks. *International Journal of Advances in Intelligent Informatics*, 2018.

- [36] Xiaoyu Liang and Taieb Znati. A long short-term memory enabled framework for ddos detection. In *2019 IEEE global communications conference (GLOBECOM)*, pages 1–6. IEEE, 2019.
- [37] Yang Luo, Qingni Shen, and Zhonghai Wu. Pml: An interpreter-based access control policy language for web services. *arXiv preprint arXiv:1903.09756*, 2019.
- [38] Tan-Khang Luong, Trung-Dung Tran, and Giang-Thanh Le. Ddos attack detection and defense in sdn based on machine learning. In *2020 7th NAFOSTED conference on information and computer science (NICS)*, pages 31–35. IEEE, 2020.
- [39] Pedro Manso, José Moura, and Carlos Serrão. Sdn-based intrusion detection system for early detection and mitigation of ddos attacks. *Information*, 10(3):106, 2019.
- [40] Tom V Mathew. Genetic algorithm. *Report submitted at IIT Bombay*, 53:18–19, 2012.
- [41] Myo Myint Oo, Sinchai Kamolphiwong, Thossaporn Kamolphiwong, and Sangsuee Vasupongayya. Advanced support vector machine-(asvm-) based detection for distributed denial of service (ddos) attack on software defined networking (sdn). *Journal of Computer Networks and Communications*, 2019(1):8012568, 2019.
- [42] Matheus P Novaes, Luiz F Carvalho, Jaime Lloret, and Mario Lemes Proença Jr. Adversarial deep learning approach detection and defense against ddos attacks in sdn environments. *Future Generation Computer Systems*, 125:156–167, 2021.
- [43] NOXRepo. Pox documentation. <https://noxrepo.github.io/pox-doc/html>, 2025. Accessed: 2025-03-04.
- [44] Jesus Arturo Perez-Diaz, Ismael Amezcua Valdovinos, Kim-Kwang Raymond Choo, and Dakai Zhu. A flexible sdn-based architecture for identifying and mitigating low-rate ddos attacks using machine learning. *IEEE Access*, 8:155859–155872, 2020.
- [45] Huseyin Polat, Onur Polat, and Aydin Cetin. Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models. *Sustainability*, 12(3):1035, 2020.
- [46] Huseyin Polat, Muammer Turkoglu, and Onur Polat. Deep network approach with stacked sparse autoencoders in detection of ddos attacks on sdn-based vanet. *IET Communications*, 14(22):4089–4100, 2020.
- [47] Rojalina Priyadarshini and Rabindra Kumar Barik. A deep learning based intelligent framework to mitigate ddos attack in fog environment. *Journal of King Saud University-Computer and Information Sciences*, 34(3):825–831, 2022.

- [48] Obaid Rahman, Mohammad Ali Gauhar Quraishi, and Chung-Horng Lung. Ddos attacks detection and mitigation in sdn using machine learning. In *2019 IEEE world congress on services (SERVICES)*, volume 2642, pages 184–189. IEEE, 2019.
- [49] M Revathi, VV Ramalingam, and B Amutha. A machine learning based detection and mitigation of the ddos attack by using sdn controller framework. *Wireless Personal Communications*, pages 1–25, 2022.
- [50] Ryu. Built-in ryu applications. https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html#get-all-flows-stats, 2025. Accessed: 2025-03-14.
- [51] Danish Sattar and Ashraf Matrawy. Towards secure slicing: Using slice isolation to mitigate ddos attacks on 5g core network slices. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 82–90. IEEE, 2019.
- [52] Qaisar Shafi, Saad Qaisar, and Abdul Basit. Software defined machine learning based anomaly detection in fog based iot network. In *Computational Science and Its Applications-ICCSA 2019: 19th International Conference, Saint Petersburg, Russia, July 1–4, 2019, Proceedings, Part IV 19*, pages 611–621. Springer, 2019.
- [53] R Shashidhara, Nisha Ahuja, M Lajuvanthi, S Akhila, Ashok Kumar Das, and Joel JPC Rodrigues. Sdn-chain: Privacy-preserving protocol for software defined networks using blockchain. *Security and Privacy*, 4(6):e178, 2021.
- [54] Yanping Shen, Kangfeng Zheng, Chunhua Wu, Mingwu Zhang, Xinxin Niu, and Yixian Yang. An ensemble method based on selection using bat algorithm for intrusion detection. *The Computer Journal*, 61(4):526–538, 2018.
- [55] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A deep learning approach to network intrusion detection. *IEEE transactions on emerging topics in computational intelligence*, 2(1):41–50, 2018.
- [56] Michael Steinbach and Pang-Ning Tan. knn: k-nearest neighbors. In *The top ten algorithms in data mining*, pages 165–176. Chapman and Hall/CRC, 2009.
- [57] Fauzi Dwi Setiawan Sumadi, Alrizal Rakhmat Widagdo, Abyan Faishal Reza, et al. Sd-honeypot integration for mitigating ddos attack using machine learning approaches. *JOIV: International Journal on Informatics Visualization*, 6(1):39–44, 2022.
- [58] Rochak Swami, Mayank Dave, and Virender Ranga. Voting-based intrusion detection framework for securing software-defined networks. *Concurrency and computation: practice and experience*, 32(24):e5927, 2020.
- [59] Naeem Firdous Syed, Syed W Shah, Arash Shaghaghi, Adnan Anwar, Zubair Baig, and Robin Doss. Zero trust architecture (zta): A comprehensive survey. *IEEE access*, 10:57143–57179, 2022.

- [60] Liang Tan, Yue Pan, Jing Wu, Jianguo Zhou, Hao Jiang, and Yuchuan Deng. A new framework for ddos attack detection and defense in sdn environment. *IEEE access*, 8:161908–161919, 2020.
- [61] Omer Elsier Tayfour and Muhammad Nadzir Marsono. Collaborative detection and mitigation of ddos in software-defined networks. *The Journal of Supercomputing*, 77(11):13166–13190, 2021.
- [62] Sahrish Khan Tayyaba, Munam Ali Shah, Omair Ahmad Khan, and Abdul Wahab Ahmed. Software defined network (sdn) based internet of things (iot): A road ahead. In *Proceedings of the International Conference on Future Networks and Distributed Systems, ICFNDS '17*, New York, NY, USA, 2017. Association for Computing Machinery.
- [63] Anurag Thantharate, Rahul Paropkari, Vijay Walunj, Cory Beard, and Poonam Kankariya. Secure5g: A deep learning framework towards a secure network slicing in 5g and beyond. In *2020 10th annual computing and communication workshop and conference (CCWC)*, pages 0852–0857. IEEE, 2020.
- [64] Wen Tian, Miao Du, Xiaopeng Ji, Guangjie Liu, Yuewei Dai, and Zhu Han. Honeypot detection strategy against advanced persistent threats in industrial internet of things: A prospect theoretic game. *IEEE Internet of Things Journal*, 8(24):17372–17381, 2021.
- [65] Nguyen Ngoc Tuan, Pham Huy Hung, Nguyen Danh Nghia, Nguyen Van Tho, Trung V Phan, and Nguyen Huu Thanh. A robust tcp-syn flood mitigation scheme using machine learning based on sdn. In *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 363–368. IEEE, 2019.
- [66] Tong Anh Tuan, Hoang Viet Long, Le Hoang Son, Raghvendra Kumar, Ishaani Priyadarshini, and Nguyen Thi Kim Son. Performance evaluation of botnet ddos attack detection using machine learning. *Evolutionary Intelligence*, 13(2):283–294, 2020.
- [67] Saif Ur Rehman, Mubashir Khaliq, Syed Ibrahim Imtiaz, Aamir Rasool, Muhammad Shafiq, Abdul Rehman Javed, Zunera Jalil, and Ali Kashif Bashir. Did-dos: An approach for detection and identification of distributed denial of service (ddos) cyberattacks using gated recurrent units (gru). *Future Generation Computer Systems*, 118:453–466, 2021.
- [68] Ruchi Vishwakarma and Ankit Kumar Jain. A honeypot with machine learning based detection framework for defending iot based botnet ddos attacks. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1019–1024. IEEE, 2019.
- [69] Jin Wang and Liping Wang. Sdn-defend: a lightweight online attack detection and mitigation system for ddos attacks in sdn. *Sensors*, 22(21):8287, 2022.

- [70] Lu Wang and Ying Liu. A ddos attack detection method based on information entropy and deep learning in sdn. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 1084–1088. IEEE, 2020.
- [71] Yang Wang, Tao Hu, Guangming Tang, Jichao Xie, and Jie Lu. Sgs: Safe-guard scheme for protecting control plane against ddos attacks in software-defined networking. *IEEE Access*, 7:34699–34710, 2019.
- [72] Yuhua Xu, Houtao Sun, Feng Xiang, and Zhixin Sun. Efficient ddos detection based on k-fknn in software defined networks. *IEEE access*, 7:160536–160545, 2019.
- [73] 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(1):9804061, 2018.
- [74] Da Yin, Lianming Zhang, and Kun Yang. A ddos attack detection and mitigation with software-defined internet of things framework. *IEEE Access*, 6:24694–24705, 2018.
- [75] Noe Marcelo Yungaicela-Naula, Cesar Vargas-Rosales, and Jesus Arturo Perez-Diaz. Sdn-based architecture for transport and application layer ddos attack detection by using machine and deep learning. *IEEE Access*, 9:108495–108512, 2021.
- [76] Kaixin Zhao, Bo Lu, Hongyu Shi, Gang Ren, and Yang Zhang. A ddos attack detection and defense mechanism based on the self-organizing mapping in sdn. *Internet Technology Letters*, 7(1):e305, 2024.
- [77] Liehuang Zhu, Md M Karim, Kashif Sharif, Chang Xu, Fan Li, Xiaojiang Du, and Mohsen Guizani. Sdn controllers: A comprehensive analysis and performance evaluation study. *ACM Computing Surveys (CSUR)*, 53(6):1–40, 2020.