

UNIVERSITY OF BAHRAIN

College of Information Technology



Design and Analysis of an Effective Architecture for Machine Learning based Networks Intrusion Detection Systems

A Thesis Submitted in Partial Fulfilment of the Requirements for the
degree of M.Sc. in Cyber Security

Submitted By

Noora Saad Alromaihi

20053019

Supervised By

Dr. Mohsen Rouached

(Associate Professor)

University of Bahrain

December 2023

ABSTRACT

Intrusion detection systems (IDS) are a crucial component within the realm of cyber security. Their primary function is the monitoring of network traffic with the aim of identifying and flagging any instances of hostile activity. In contemporary times, the efficacy of conventional Intrusion Detection Systems (IDS) has been called into question due to the exponential proliferation of internet use, hence giving rise to a surge in novel forms of cyber assaults. Machine learning (ML) techniques have emerged as a viable alternative for augmenting conventional intrusion detection systems (IDS). This study will examine the variations in traffic received by Network Intrusion Detection Systems (NIDS) due to their installation in distinct subnetworks within an organization. In several instances, it is seen that a single assailant may direct their efforts towards different servers belonging to various organizations. This behavior is often perceived by intrusion detection systems (IDS) as infrequent attacks, hence diminishing the effectiveness of detection. The objective is to develop a machine learning-driven IDS model capable of identifying and flagging potentially harmful network traffic received across various interfaces inside organizational networks. The suggested approach involves the establishment of a centralized proxy server that will act as the recipient for all incoming traffic directed toward the organization's servers. This proxy server will be equipped with a multi-model IDS that is designed to analyze the traffic load. Upon analysis, the proxy server will then redirect the traffic to the appropriate requested server. The project has two primary objectives: the construction of an architecture that guarantees a comprehensive perspective and the examination of incoming traffic directed toward the servers of various organizations. The development of a machine-learning anomaly-based IDS with a high level of accuracy and detection rate. The ML-based IDS under consideration will be evaluated alongside other models suggested in existing literature studies, with a focus on assessing their respective accuracy percentages and detection rates. The evaluation of the proposed IDS included the use of three distinct datasets, namely the CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017 datasets. The XGBoost model demonstrated exceptional performance in terms of rapid detection, achieving accuracy rates of 99.96%, 99.73%, and 99.84% for the CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017 datasets, respectively. These impressive results were obtained within very short time intervals of 0.01 seconds, 0.03 seconds, and 0.04 seconds, respectively. The Stacking model has the highest level of accuracy among the evaluated models. Specifically, it achieves an accuracy of 99.98% for the CIC-MalMem-2022 dataset, 99.61% for the CIC-IDS-2018 dataset, and 99.86% for the CIC-IDS-2017 dataset. The IDS that was developed

demonstrated superior accuracy and detection time outcomes in comparison to previous research in the field.

Contents

APPROVAL	A
DECLARATION	B
ABSTRACT	C
LIST OF FIGURES	I
LIST OF TABLES	M
DEDICATION	O
LIST OF ABBREVIATIONS	P
1 Introduction	1
1.1 Overview	1
1.2 Background	1
1.3 Problem Definition	2
1.4 Research Objectives and Questions	3
1.5 Research Significance	4
1.6 Research Contributions	4
1.7 Research Methodology	5
1.8 Thesis Structure	6
1.9 Summary	7
2 Literature Review	8
2.1 Overview	8
2.2 Intrusion Detection Systems	8
2.2.1 Data Source-Based Methods	9

2.2.2	Detection-Based Methods	11
2.3	Machine Learning	12
2.3.1	Advantages and Disadvantages of ML Family . . .	14
2.3.1.1	Information-based Learning Techniques .	14
2.3.1.2	Similarity-Based Learning Techniques . .	16
2.3.1.3	Probability-based Learning Techniques . .	17
2.3.1.4	Error-based Learning Techniques	18
2.4	Related Work	19
2.4.1	Information-based Learning Techniques	19
2.4.2	Similarity-Based Learning Techniques	21
2.4.3	Probability-based Learning Techniques	22
2.4.4	Error-based Learning Techniques	23
2.4.5	Hybrid Model	24
2.4.6	SDN Model	28
2.5	Discussion	30
2.6	Summary	35
3	Methodology	36
3.1	Overview	36
3.2	Multi-model IDS based on traffic load	36
3.3	Benchmark Dataset	43
3.3.1	CIC-MalMem-2022	44
3.3.2	CIC-IDS-2018	44
3.3.3	CIC-IDS-2017	45
3.4	Features Selection	47
3.4.1	Univariate Feature Selection	48
3.4.2	Removing Features with Low Variance	48
3.5	Data Preparation	49
3.5.0.1	CIC-MalMem-2022	49
3.5.0.2	CIC-IDS-2018	50

3.5.0.3	CIC-IDS-2017	50
3.6	Machine Learning Models	58
3.6.1	AdaBoost	58
3.6.2	Voting Classifier	58
3.6.3	Stacking Classifier	59
3.6.4	Support Vector Machine	59
3.6.5	Random Forest	60
3.6.6	XGBoost	60
3.7	Performance Evaluation	60
3.7.1	Confusion Matrix	61
3.7.2	Learning curve	62
3.7.3	Hyper-parameter in the Model	63
3.8	Summary	65
4	Results and Discussion	66
4.1	Overview	66
4.2	Results	66
4.2.1	AdaBoost	66
4.2.1.1	Learning Curve	68
4.2.1.2	Hyperparametr in the Model	69
4.2.2	Voting Classifier	71
4.2.2.1	Learning Curve	73
4.2.2.2	Hyperparametr in the Model	74
4.2.3	Stacking Classifier	75
4.2.3.1	Learning Curve	77
4.2.3.2	Hyperparametr in the Model	78
4.2.4	SVM	79
4.2.4.1	Learning Curve	81
4.2.4.2	Hyperparametr in the Model	82
4.2.5	Random Forest	83

4.2.5.1	Learning Curve	85
4.2.5.2	Hyperparametr in the Model	86
4.2.6	XGBoost	88
4.2.6.1	Learning Curve	90
4.2.6.2	Hyperparametr in the Model	91
4.3	Discussion	92
4.4	Limitations of Proposed Model	95
4.5	Summary	96
5	Conclusion and Recommendations	98
5.1	Overview	98
5.2	Conclusion	98
5.3	Recommendations and Future Work	101
	References	102
	Abstract (in Arabic)	

List of Figures

1.1	Methodology and process of the study	6
2.1	Taxonomy of IDSs	10
2.2	IDS challenges and detection methods	13
3.1	Multi-model based network traffic load	41
3.2	Machine learning model architecture	47
3.3	Binary class CIC-MalMem-2022 dataset	52
3.4	Binary class for CIC-IDS-2018 dataset	52
3.5	Binary class for CIC-IDS-2017 dataset	52
4.1	Confusion matrix on the CIC-MalMem-2022 dataset for Ad- aBoost	67
4.2	Confusion matrix on the CIC-IDS-2018 dataset for AdaBoost	67
4.3	Confusion matrix on the CIC-IDS-2017 dataset for AdaBoost	67
4.4	Learning Curve on the CIC-MalMem-2022 dataset for Ad- aBoost	69
4.5	Learning Curve on the CIC-IDS-2018 dataset for AdaBoost .	69
4.6	Learning Curve on the CIC-IDS-2017 dataset for AdaBoost .	69
4.7	Hyperparameter tuning on the CIC-MalMem-2022 dataset for AdaBoost	70
4.8	Hyperparameter tuning on the CIC-IDS-2018 dataset dataset for AdaBoost	70

4.9	Hyperparameter tuning on the CIC-IDS-2017 dataset dataset for AdaBoost	70
4.10	Confusion matrix on the CIC-MalMem-2022 for Voting classifier	72
4.11	Confusion matrix on the CIC-IDS-2018 dataset for Voting classifier	72
4.12	Confusion matrix on the CIC-IDS-2017 dataset for Voting classifier	72
4.13	Learning curve on the CIC-MalMem-2022 dataset for Voting classifier	74
4.14	Learning curve on the CIC-IDS-2018 dataset for Voting classifier	74
4.15	Learning curve on the CIC-IDS-2017 dataset for Voting classifier	74
4.16	Confusion matrix on the CIC-MalMem-2022 for Stacking classifier	76
4.17	Confusion matrix on the CIC-IDS-2018 dataset for Stacking classifier	76
4.18	Confusion matrix on the CIC-IDS-2017 dataset for Stacking classifier	76
4.19	Learning curve on the CIC-MalMem-2022 dataset for Stacking classifier	78
4.20	Learning curve on the CIC-IDS-2018 dataset for Stacking classifier	78
4.21	Learning curve on the CIC-IDS-2017 dataset for Stacking classifier	78
4.22	Confusion matrix on the CIC-MalMem-2022 dataset for SVM	
	80	
4.23	Confusion matrix on the CIC-IDS-2018 dataset for SVM . .	80

4.24	Confusion matrix on the CIC-IDS-2017 dataset for SVM . . .	80
4.25	Learning curve on the CIC-MalMem-2022 dataset for SVM	82
4.26	Learning curve on the CIC-IDS-2018 dataset for SVM . . .	82
4.27	Learning curve on the CIC-IDS-2017 dataset for SVM . . .	82
4.28	Hyperparametr tuning on the CIC-MalMem-2022 dataset for SVM	83
4.29	Hyperparametr tuning on the CIC-IDS-2018 dataset for SVM	83
4.30	Hyperparametr tuning on the CIC-IDS-2017 dataset for SVM	83
4.31	Confusion matrix on the CIC-MalMem-2022 dataset for RF	84
4.32	Confusion matrix on the CIC-IDS-2018 dataset for RF . . .	84
4.33	Confusion matrix on the CIC-IDS-2017 dataset for RF . . .	84
4.34	Learning curve on the CIC-MalMem-2022 dataset for RF . .	86
4.35	Learning curve on the CIC-IDS-2018 dataset for RF	86
4.36	Learning curve on the CIC-IDS-2017 dataset for RF	86
4.37	Hyperparametr tuning on the CIC-MalMem-2022 dataset for RF	88
4.38	Hyperparametr tuning on the CIC-IDS-2018 dataset for RF .	88
4.39	Hyperparametr tuning on the CIC-IDS-2017 dataset for RF .	88
4.40	Confusion matrix on the CIC-MalMem-2022 dataset for XG- Boost	89
4.41	Confusion matrix on the CIC-IDS-2018 dataset for XGBoost 89	
4.42	Confusion matrix on the CIC-IDS-2017 dataset for XGBoost 89	
4.43	Learning curve on the CIC-MalMem-2022 dataset for XG- Boost	91
4.44	Learning curve on the CIC-IDS-2018 dataset for XGBoost .	91
4.45	Learning curve on the CIC-IDS-2017 dataset for XGBoost .	91

4.46	Hyperparametr tuning on the CIC-MalMem-2022 dataset for XGBoost	92
4.47	Hyperparametr tuning on the CIC-IDS-2018 dataset for XG- Boost	92
4.48	Hyperparametr tuning on the CIC-IDS-2017 dataset for XG- Boost	92

List of Tables

2.1	An overview of the ML family and its algorithms	14
2.2	Summary of LR	33
3.1	A comparison of the critical differences between centralized and decentralized network architectures	39
3.2	Comparing the advantages and disadvantages of centralized and decentralized network architectures	39
3.3	A comparison of the CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017 datasets	45
3.4	Features for CIC-MalMem-2022 dataset.	54
3.5	Features for CIC-IDS-2017 dataset	55
3.6	Some of Features for CIC-IDS-2018 dataset.	56
3.7	A comparison of the CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017 datasets after preprocessing steps	57
3.8	A confusion matrix for binary classifications	61
3.9	Hyper-parameter and tuning values for each ML algorithm .	65
4.1	Summary of Adaboost model results	68
4.2	AdaBoost hyperparameter results for CIC-MalMem-2022 dataset	70
4.3	Summary of Voting classifier model results	72
4.4	Voting hyperparameter results for CIC-MalMem-2022, CIC- IDS-2018 ,and CIC-IDS-2017 datasets	75

4.5	Summary of Stacking classifier model results	76
4.6	Stacking classifier hyperparameter results for CIC-MalMem- 2022,CIC- CIC-IDS-2018, CIC-IDS-2017 datasets	79
4.7	Summary of SVM model results	81
4.8	Summary of Random Forest model results	85
4.9	Summary of XGBoost model results	90
4.10	An analysis of the CIC-MalMem-2022 dataset for each model 93	
4.11	An analysis of the CIC-IDS-2018 dataset for each model . .	93
4.12	An analysis of the CIC-IDS-2017 dataset for each model . .	93
4.13	Comparative analysis of the CIC-MalMem-2022 dataset . .	94
4.14	Comparative analysis of the CIC-IDS-2018 dataset	94
4.15	Comparative analysis of the CIC-IDS-2017 dataset	95

DEDICATION

As part of my academic journey, I am thankful for Dr. Mohsen Rouached's kind cooperation and invaluable assistance. Furthermore, I would like to thank all the professors who have taught me over the past two years, especially Dr. Hessa Jassim Al-Junaid and Dr. Wael Mohamed Elmedany. I am also grateful for my supportive family's constant encouragement and unwavering support. Finally, I thank my friends and colleagues who have provided invaluable assistance and support throughout my academic career.

LIST OF ABBREVIATIONS

IDSs	Intrusion Detection Systems
HIDS	Host-based IDSs
NIDS	Network-based IDSs
ML	Machine Learning
DL	Deep Learning
RF	Random Forest
CNN	Convolutional Neural Network
BiLSTMs	Bidirectional LSTM
DNN	Deep Neural Networks
LSTM	Long Short-Term Memory Networks
t-SNE	t-Distributed Stochastic Neighbor Embedding
SDN	Software-Defined Networking
CC	Cloud Computing
IoT	Internet of Things
SMOT	Synthetic Minority Oversampling Technique
XG-Boos	Extreme Gradient Boosting
PCA	Principal Component Analysis
DCNN	Deep Convolutional Neural Network

NSBPSO Novel Neighborhood Search-based Particle Swarm Optimization

EGA-PSO Enhanced Genetic Algorithm and Particle Swarm Optimization

IRF Improved Random Forest

ANOVA Analysis of variance

TACHAN Text Conditioned Auxiliary Classifier Generative Adversarial
Network

DAN Analysis of variance

DSC Discrete Singular Convolution

SVM Support Vector Machine

CGO Chaos Game Optimization

MDS Multidimensional Scaling

CAE Classical Auto-encoder

RFE Recursive Feature Elimination

DT Decision Trees

Cu-Blstm Cubic Long Short-Term Memory

GA Genetic Algorithm

MLP Multilayer Perceptron

FAR False Alarm Rate

LDR Low Detection Rate

WAN Wide Area Network

LAN Virtual Local Area Network

TP True Positive

TN True Negative

FP False Positive

FN False Negative

Chapter 1

Introduction

1.1 Overview

This chapter provides an overview of the research idea, the questions to be addressed, and the research methodology. It also discusses the research's contributions.

1.2 Background

In the realm of online security, the predominant concern is in the occurrence of security breaches, which mostly stem from the act of intrusion. In the event of a single error or unauthorized access, it is possible for an individual to swiftly gain control over or erase data pertaining to the computer and system architecture. In addition, a centralized network design may exhibit increased susceptibility to several forms of security breaches, including unauthorized access, data breaches, and distributed denial-of-service (DDoS) assaults. Furthermore, the occurrence of an intrusion may lead to substantial financial ramifications and compromised computer transactions, ultimately leading to a dearth of data in the context of a cyber digital conflict. The system may be negatively impacted by security failures (Azizan et al., 2021), (Priyadarsini & Bera, 2021). According to (Samantaray, Satapathy,

& Lenka, 2022), the implementation of intrusion detection systems is crucial for mitigating failures. In a broad sense, an intrusion detection system (IDS) is designed to identify and respond to network assaults by actively monitoring the software and hardware configurations of a given network. By doing so, it plays a crucial role in safeguarding against cyber threats (Soomro, Naeem, Ghafoor, Senapati, & Rajwana, 2023). There exist many challenges linked with Intrusion Detection Systems (IDS), including the incapability to identify novel assaults, a very low rate of accurate detection, and the need to mitigate false alarm occurrences. Furthermore, it is worth noting that anomalous intrusion detection approaches are still facing ongoing challenges in terms of achieving consistent performance improvements (Jaradat, Barhoush, & Easa, 2022). The use of machine learning and deep learning methodologies is often seen in the context of augmenting the efficacy of intrusion detection systems (Jaradat et al., 2022). As an advanced technological system with the ability to access and retrieve. Due to its ability to extract important information from vast datasets in an automated manner, machine learning (ML) has garnered significant attention in academic circles. This is mostly due to its effectiveness as a tool for knowledge retrieval. By using an ample amount of training data, Intrusion Detection Systems (IDSs) may attain high levels of perceptiveness, while machine learning models demonstrate satisfactory generalization capabilities in the realm of attack detection. One further benefit of machine learning is its ease of design and development, as highlighted by (Amanoul, Abdulazeez, Zeebare, & Ahmed, 2021).

1.3 Problem Definition

Ensuring network security for an organization is of utmost importance due to the widespread use of the Internet for inter-organizational and inter-company

communication, which amplifies the potential for the dissemination of attacks and threats among these entities. Conventional intrusion detection systems may be inadequate in detecting novel infiltration attempts. Moreover, the Network intrusion detection system (NIDS) receives different traffic since every IDS is installed in other subnetworks of the organization. In many cases, the same attacker may target multiple organization servers seen by each IDS as seldom attack, decreasing the detection rate. Therefore, this situation poses challenges in accurately discerning genuine dangers and promptly implementing appropriate measures. Organizations have to use more advanced solutions, such as intrusion detection systems based on machine learning, in order to enhance their security stance.

1.4 Research Objectives and Questions

The idea is to create a machine learning-based IDS model to detect malicious traffic received by different organization network interfaces. By creating a centralized proxy server to receive all the traffic from the organization's servers, scan the traffic using our proposed IDS, and then redirect the traffic to the requested server. So, the project has two objectives:

1. Design of an architecture ensuring the global view and inspection of the traffic coming to the different organization servers.
2. Creation of machine-learning anomaly-based IDS with high accuracy and detection rate, and Intrusion Prevention Systems (IPS) prevention of malicious IPs through firewalls.

The research questions for this study are as follows:

1. What architecture is most suitable for meeting the needs of an efficient and effective IDS?

2. How can centralized network architecture enhance the functionality of intrusion detection systems?
3. How does using machine learning algorithms in intrusion detection systems help protect against intrusions?

1.5 Research Significance

The significance of the research focuses on enhancing traditional IDS with machine learning solutions to provide efficient security solutions for networks in organizations and companies. Additionally, it presents a network strategy that effectively addresses security breaches. Moreover, machine learning can detect and respond to cyber threats more accurately and rapidly than traditional methods, reducing the time and effort involved in detecting security breaches. As a result, an organization's cyber security posture would be enhanced, and the risk of a successful cyberattack would be reduced.

1.6 Research Contributions

Contributions of the thesis include the following:

1. Propose a novel centralized network architecture based on multi-model IDS based on traffic load. The model is more realistic since the organizations have a prime time or specific events.
2. In the proposed model, global traffic inspection occurs before traffic is sent to the central server and other sub-organizations.
3. Analyze the results of Adaboost, Voting, Stacking, Support Vector

Machine, Random Forest, and XGBoost models on the CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS- 2017 datasets.

4. Comparison of the best models with existing IDSs.

1.7 Research Methodology

This study used a quantitative and exploratory methodology. The literature review included an examination of many research articles and theses. In this work, three cybersecurity datasets were employed, namely CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017. Machine learning models are developed at Google Colab using the Python programming language, with the SciKit Learn package being of paramount significance in their creation. Prior to incorporating the dataset into the model, it is essential to engage in re-engineering processes to ensure its compatibility and alignment with the model. The models used for evaluating the dataset encompass AdaBoost, Voting, Stacking, SVM, Random Forest, and XGBoost. The following section presents a novel approach for developing a multi-model intrusion detection system that leverages traffic load as a key factor in a centralized network. Given that businesses often have prime-time or specialized events, the suggested network design presents a more realistic approach and provides enhanced security across many layers. The initial layer of security involves the implementation of a traffic monitoring system at the organization's Wide Area Network (WAN) entry point. If the incoming traffic exceeds a predetermined threshold established by the network administrator, the study will employ the fastest detection model, which in this case is XGBoost. Conversely, if the traffic remains below the threshold, the study will utilize a stacking classifier model known for its high accuracy. The second layer of security involves the distribution of traffic to sub-organizations, where the deployed Intrusion Detection Systems (IDS) further analyze the traffic using

a highly accurate model known as Stacking. An overview of the research methodology for this study can be found in Figure 1.1

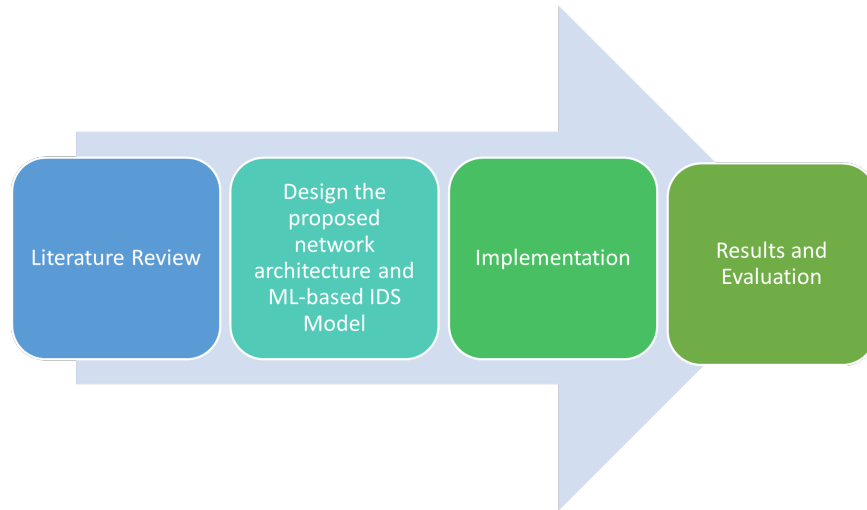


Figure 1.1: Methodology and process of the study

1.8 Thesis Structure

Following is a brief outline of the rest of the thesis:

- Chapter 2: This chapter discusses previous work on network intrusion detection enhanced with machine learning. Also highlight the main point in the IDS and ML.
- Chapter 3: Describe the novel centralized network architecture, datasets used in the experiment, and machine learning models that were used in the experiment.
- Chapter 4: Conduct an analysis and presentation of the experiment's results and compare them with those obtained from previous studies.
- Chapter 5: This final chapter discusses the conclusions and recommendations, summarizes the work done in this investigation, and finally highlights recommendations for further research.

1.9 Summary

This introduction aims to provide an overview of centralized networks, IDS, and ML, as well as to describe the problem definition, the research objectives, the research importance, the research contribution, and the research questions.

Chapter 2

Literature Review

2.1 Overview

In cyber security, detection accuracy is the primary challenge for machine learning-based intrusion detection systems (IDS). This literature review aims to develop an innovative architecture for machine learning-based IDS.

2.2 Intrusion Detection Systems

Intrusion detection systems (IDS) are used to monitor network traffic behavior or analyze data generated by a host in order to detect and identify any unauthorized entry attempts that compromise security. These technologies serve as the primary means of protection for the industry against possible cybersecurity attacks. Upon the discovery of an assault, an alarm is promptly sent. The core principles of IDS are centered on the actions of monitoring, detecting, and responding to any instances of unauthorized ac-

tivity. An intrusion refers to the unauthorized use, exploitation, and manipulation of computer systems by individuals, both external and internal, in a manner that is not sanctioned. To safeguard a computer system or computer network, it is essential for an Intrusion Detection System (IDS) to first detect and recognize malicious assaults targeting host devices or network systems (Verma & Ranga, 2020). The primary objective of an Intrusion Detection System (IDS) is to monitor network events and identify any instances of aberrant behavior. Intrusion Detection Systems (IDS) are responsible for capturing data pertaining to identified security incidents and then delivering this information to administrators in charge of security. In addition to their primary function of detecting security policy breaches, IDSs are also used for the purpose of reporting existing dangers and dissuading persons from engaging in activities that contravene security regulations (Aljanabi, Ismail, & Ali, 2021), (Ozkan-Okay, Samet, Aslan, & Gupta, 2021). There are two categorization approaches that may be used in Intrusion Detection Systems (IDS), which are distinguished depending on the data source and detection (Mirlekar & Kanojia, 2022). Each approach has distinct characteristics. There are two distinct groups of data source-based methods: host-based Intrusion Detection Systems (HIDS) and network-based IDSs(NIDS). In addition, it is worth noting that there are two distinct categories of detection-based approaches, namely anomaly-based and signature-based detection (Unnisa A, Yerva, & MZ, 2022). Figure 2.1 illustrates the classification of IDSs.

2.2.1 Data Source-Based Methods

According (Abubakar & Garko, 2021), it has been noted that Host-based Intrusion Detection Systems (HIDS) have the capability to identify insider attacks that are not directly associated with network traffic. Furthermore, HIDS does an analysis of data derived from the host system, including log

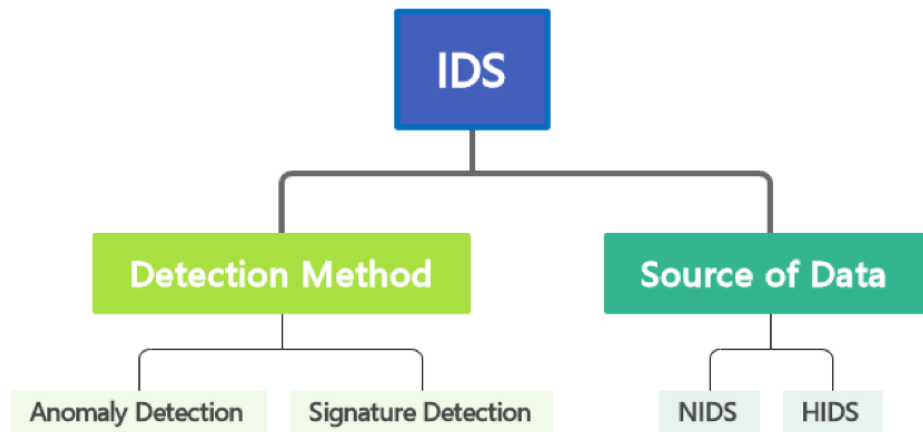


Figure 2.1: Taxonomy of IDSs

files originating from databases, firewalls, operating systems, and Windows servers. Host-based intrusion detection systems (IDSs) offer numerous benefits. These include the ability to configure them in a flexible manner, enabling finer monitoring granularity. Additionally, host-based IDSs do not necessitate the use of additional hardware, allowing for cost savings. They also facilitate the determination of attack success, as well as the utilization of the environment for encryption and exchange purposes. Moreover, host-based IDSs ensure that they do not disrupt the network's normal functioning. Nevertheless, HIDS exhibits a number of drawbacks in an academic context. Firstly, their cost is higher compared to network-based IDS (NIDS) due to the need to obtain a detector license for each server. Additionally, host-based IDS use a substantial percentage of the server's CPU capacity, thus impacting its overall performance. Furthermore, these systems demand a specific allocation of memory and hard drive space, further adding to their resource requirements. In the event that the detector is operating inside an overloaded system, so impacting the verification perfor-

mance, it becomes essential to get a terminal equipment room that offers enhanced performance (H. Zhang, Lan, & Zhang, 2022). The NIDS offers the functionality of monitoring network traffic via the extraction of various network data, including NetFlow, packet capture, and other relevant information. There are several data sources accessible on the network. The Network Intrusion Detection System (NIDS) has the functionality to monitor both internal computers inside a network and detect exterior hostile activities (Abubakar & Garko, 2021). NIDS has several benefits, such as prompt detection and a reduced number of detectors; nevertheless, it is important to note that the coverage area is extensive. The NIDS system exhibits independence from the underlying operating system, hence enabling its use across diverse operating systems. One primary limitation of NIDS is their inherent capability to just monitor the network traffic that traverses a designated area of the network. The notification system of network intrusion detection (NIDS) will only alert the user when the communication data adheres to the predetermined criteria and characteristics. It is important to note that NIDS is unable to access encrypted communication data. Additionally, certain configurations are required for switch networks to enable access to all communication data (H. Zhang et al., 2022).

2.2.2 Detection-Based Methods

The approaches referred to as signature-based methods are often recognized as misuse detection methods (Abubakar & Garko, 2021). This approach involves the comparison of the signature of a given sample with the signature stored in the database. This approach has several benefits, including a minimal incidence of false alarms and comprehensive documentation of the nature and causation of the assaults. Furthermore, this approach exhibits a notable degree of detection efficacy. One notable drawback of this approach is its inability to identify novel threats (Unnisa A et al., 2022). The second

approach is anomaly-based identification, which aims to identify deviations in the behavior of the profile. The aforementioned approach offers notable benefits due to its effectiveness in identifying novel assaults and its ability to optimize performance based on the intricacy of the model. One limitation of this approach is its elevated risk of false alarms and its limited interpretability since it just presents observed outcomes (Unnisa A et al., 2022).

After thoroughly examining the benefits and drawbacks of Data-Source-Based Methods and Detection-Based Methods, it has been determined that Intrusion Detection Systems (IDS) pose four notable challenges (Aljanabi et al., 2021):

- Low detection rate.
- False alarm rate.
- Response time.
- Unbalanced dataset.

In order to overcome these challenges, (Alajanbi, Ismail, Hasan, & Sulaiman, 2021) the following detection methods can be used with IDS:

- Machine Learning.
- Data Mining.
- Statistical Methods.

Figure 2.2 IDS challenges and detection methods.

2.3 Machine Learning

The significance of integrating machine learning (ML) into the field of cybersecurity has escalated due to its ability to rapidly assess and detect various forms of threats from vast quantities of events (Yan, Wen, Nepal, Paris,

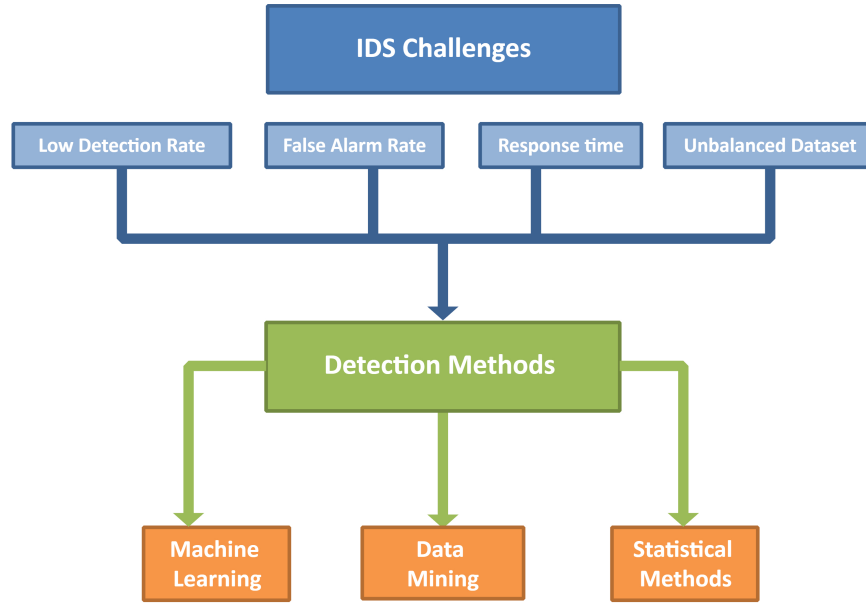


Figure 2.2: IDS challenges and detection methods

& Xiang, 2022). Moreover, ML is a methodology used to automate the procedure of constructing analytical models using data, whereby the models acquire the ability to make judgments via learning (Jahwar & Ameen, 2021). There are two primary categories of machine learning algorithms, with the first category consisting of shallow models. The second approach under consideration is deep learning, as mentioned by (Ramanan, 2022). According (Sarker, 2021), the categorization of learning algorithms is based on the manner in which data is used, resulting in four distinct types: unsupervised, semi-supervised, supervised, and reinforcement learning. In contrast, there exist four distinct categories of machine learning (ML) that possess independent learning characteristics. These categories include information-based learning techniques, similarity-based learning techniques, probability-based learning techniques, and error-based learning techniques. Collectively, these

categories are commonly referred to as the ML family (Kostinakis, 2022), (Kelleher, Mac Namee, & D’arcy, 2020). Table 2.1 presents a comprehensive overview of the ML family and its corresponding algorithms.

Table 2.1: An overview of the ML family and its algorithms

ML Family	Description	Algorithms
Information-based Learning Techniques	Building models based on information theory concepts.	Random Tree, Random Forest, J48, HoeffdingTree, REPTree , and Decision Stump.
Similarity-Based Learning Techniques	Comparison of known and unknown objects or measuring the degree of similarity between past and forthcoming events can be used to develop a model.	k-NN and k-means (simple k-means, Canopy, Hierarchical)
Probability-based Learning Techniques	Developing a model based on measuring the probability of an event occurring.	Naïve Bayes
Error-based Learning Techniques	Utilizing a set of training instances to build a model that minimizes the total error.	SVM, Logistics, Perceptron, Winnow, and Deep Learning.

2.3.1 Advantages and Disadvantages of ML Family

This part discusses the advantages and disadvantages of each ML family.

2.3.1.1 Information-based Learning Techniques

Through exploration and challenging questions, researchers are involved in information-based learning, which is a method of teaching that makes connections to the real world (Sawant & Prabukumar, 2020). There are several

advantages and disadvantages associated with information-based learning techniques, such as (Kelleher et al., 2020):

Advantages:

1. **Interpretable:** Among the many significant advantages of decision trees is their intuitive and straightforward understanding. In addition, decision trees can be utilized by data scientists and other professionals to communicate the model's predictions in a flowchart-like manner to stakeholders. By providing detailed information about the situation and the reasons for it, stakeholders will gain confidence.
2. As well as modeling interactions between descriptive features, a decision tree approach also provides a mechanism for modeling interactions between defining features and interactions between defining features. Therefore, the tree nodes are evaluated by comparing them to the evaluations of other descriptive features conducted at the preceding nodes on the path from the root. Decision trees are thus a valuable tool for modeling interaction between descriptive features. As a result of pre-pruning, descriptive feature interactions may be less easily captured since no subtrees may emerge that capture descriptive feature interactions.

Disadvantages:

1. A decision tree has the disadvantage that it is a high-variance algorithm that is prone to over-fitting since it cannot stop automatically, thus resulting in decisions that are difficult and complex to make.
2. Considering decision trees are eager learners, they are unsuitable for modeling concepts that change over time since they must be retrained every time.

3. As a general rule, decision trees are appropriate for dealing with continuous and categorical features. However, when dealing with continuous descriptive features, the tree may grow quite large, making it difficult to interpret the tree. Other prediction models may be more suitable for continuous data.

2.3.1.2 Similarity-Based Learning Techniques

AI transparency and trust are offered through similarity-based machine learning. In order to determine how similar two or more items are to one another based on algorithmic distance functions, the similarity machine learning method employs a nearest-neighbor approach. Recommendation systems face recognition or face verification and information retrieval for learning to rank all use similarity learning (Shrivastava & Sisodia, 2019). Additionally, many approaches to machine learning depend on a certain metric. It has been analyzed that unsupervised learning techniques like clustering, which compiles related or nearby objects, are included in this (Nagaraja et al., 2020). There are several advantages and disadvantages associated with similarity-based learning techniques, such as (Kelleher et al., 2020):

Advantages:

1. There are several strengths to these models, including their ease of interpretation, their ability to handle a wide range of descriptive features, their relative robustness to noise (when k is appropriately set), and their ability to handle concept drift may be stronger than those induced by eager learning algorithms.

Disadvantages:

1. In addition to their sensitivity to dimensionality, similarity-based learning approaches may have a slow prediction speed, mainly when work-

ing with massive datasets, and their lack of accuracy comparable to other approaches.

2.3.1.3 Probability-based Learning Techniques

Probabilistic classifiers are classifiers that, rather than producing a most likely class for observation, can predict a probability distribution of all classes within that collection based on an observation of an input. ML's fundamental concept is probability, which expresses how likely an event is to occur. The probability's value is always between 0 and 1. In order to comprehend ML models and their applications, one must first understand this fundamental idea (Y. Zhao & Wang, 2022). There are several advantages and disadvantages associated with probability-based learning techniques, such as (Kelleher et al., 2020):

Advantages:

1. The ability to fully comprehend the uncertainty related to forecasts is one of the probabilistic modeling technique's key benefits. By employing this technique, it can immediately ascertain the level of assurance and forecast accuracy of any mobile learning model (Modi & Swain, 2023).
2. It saves a lot of time and works quickly.
3. It is more suitable for categorical than numerical input variables.

Disadvantages:

1. One of the drawbacks of the probabilistic modeling approach is that it necessitates thorough development, a process that is dependent on numerous assumptions and a significant amount of input data (Modi & Swain, 2023).

2. Since the algorithm assumes that all predictors or features are independent, it cannot be used for real-world situations.

2.3.1.4 Error-based Learning Techniques

Learning with error-based approaches involves identifying, analyzing, and correcting one's own mistakes as one performs tasks through methodical feedback approaches. This learning strategy uses linear regression, error measurement, and error surfaces. Multivariate linear regression and gradient descent are the main components of this learning strategy (Ajila, Lung, & Das, 2021). There are several advantages and disadvantages associated with error-based learning techniques, such as (Kelleher et al., 2020):

Advantages:

1. As opposed to only having a broad comprehension of the idea, the learner now has the opportunity to acquire "depth" in that understanding (Ramirez-Hernandez, Stolwyk, Ownsworth, & Wong, 2021).
2. Various relationships between variables can be modeled using this method.
3. In addition to providing greater variability and less collinearity among variables, panel data also provides greater degrees of freedom.

Disadvantages:

1. In general, a basis function model has a much greater number of weights than there are defining features; therefore, choosing the optimal weights requires searching a much broader range of possibilities.
2. To fully understand a learner's competency, it is necessary to look into both errors and non-errors. Some significant aspects of learners'

competence, such as the structures they avoid, may not be obvious because error analysis only looks at learners' production (Ramirez-Hernandez et al., 2021).

3. Using panel data models has limitations, such as heterogeneity, correlation in disturbance terms, and heteroscedasticity. There may be behavioral differences among their elements, which must be considered when conducting the analysis. They may behave similarly to other groups but differently from those of other groups.

2.4 Related Work

This section discusses several recent research papers that investigate machine learning-based intrusion detection systems and deep learning-based intrusion detection systems.

2.4.1 Information-based Learning Techniques

The proposed enhanced IDS employs machine learning algorithms (Attou, Guezaz, Benkirane, Azrour, & Farhaoui, 2023) that monitor resources, services, and networks and then detect attacks on the cloud based on the monitoring data. As a result of testing and training the proposed model using the BoT-IoT dataset, which includes RF algorithms for predicting and detecting intrusions and graphic visualization for feature engineering, it has achieved a 99.99 percent accuracy rate. Several limitations are present due to this study because several cloud computing deployment models have yet to be examined (public, private, hybrid), and the Bot-IoT and NSL-KDD datasets have been used solely for evaluation. A third issue in this study was the need to compare various machine learning algorithms concerning the system's performance. As a result of machine learning methods and feature dimen-

sional reduction,by (Saheed, Abiodun, Misra, Holone, & Colomo-Palacios, 2022) developed the IoT IDS to detect abnormal behavior on an IoT network. The UNSWNB15 dataset was prevented from leaking information by a feature scaling method, the min-max normalization method, followed by a PCA that reduced dimensionality. In order to conduct the analysis, the dataset was collected and loaded, and the first step was to preprocess the data. Through the preprocessing process, PCA is used to select the most relevant features from the data by eliminating redundant attributes and outliers based on the results of the min-max algorithm. The Min-Max method was employed to normalize the data, followed by a PCA analysis that selected ten critical attributes from the 49 attributes in the dataset. After trimming the dataset, XGBoost was trained on it. As part of this study, the system's performance has only been evaluated using the UNSW-NB15 dataset, but other typologies of IoT networks have yet to be evaluated. (Seth, Singh, & Kaur Chahal, 2021) suggest that a hybrid feature selection approach can reduce model complexity, prediction latency, and attack prediction performance without adversely affecting the model. The authors use a lightweight gradient boosting framework called LightGBM (LightGBM) to construct a model from the CIC-IDS 2018 dataset. The proposed feature selection is estimated to have a prediction latency of 45.52 percent to 2.25 percent, whereas the proposed model is estimated to have a construction time of 52.68 percent to 17.94 percent. The selected features were selected using RF and PCA techniques. LightGBM combined with hybrid feature selection resulted in 97.73 percent accuracy, 99.3 percent sensitivity, 99.3 percent precision, and a relatively short prediction delay. The proposed model demonstrated a 1.5 percent improvement in accuracy and a 3 percent improvement in precision compared with the existing model. To provide a comprehensive understanding of how network parameters vary during benign or malicious sessions, a detailed analysis of network parameters is conducted in this

study. This study utilized a single dataset and machine-learning algorithm; however, several areas for improvement exist.

2.4.2 Similarity-Based Learning Techniques

A solution to the problem of sparse data has been proposed by (Ding, Chen, Dong, Fu, & Cui, 2022) to minimize the impact of sparse data on detection performance by utilizing a DAN method to extract features from the data. Data of minority classes within overlapped regions is created using TACGAN. By utilizing information loss, minority class data is more consistent with the sampling distribution of the original data used to generate the TACGAN model generator data. The data filtering module removes noise from the samples as soon as the TACGAN model has been completed. For KDDCUP99, UNSW-NB15, and CICIDS2017, accuracy results were 93.53%, 92.39%, and 95.86%, respectively. This study did not evaluate the algorithm's performance by using different hyperparameters. By utilizing K-nearest neighbors and artificial neural networks (ANNs), the application has been developed by (Dini & Saponara, 2021) to analyze the traffic flow of local area networks. In order to verify the audit data by simulating intrusions in an environment similar to a defense network environment, a US Air Force emulated LAN was utilized to obtain authenticated TCP/IP dump data. In contrast to PCA, this technique has performed better by selecting the cosine of similarity to compare observations in the training set using a feature reduction approach based on multidimensional scaling. The datasets examined in this study, including typical LAN traffic used by the US Air Force, accurately represented LAN traffic in military applications. The accuracy of the ANN was found to be 99.23 percent, while the accuracy of the KNN was found to be 99.57 percent. There are several issues with current intrusion detection systems, including false positives and low detection rates. Furthermore, the intrusion detection system cannot detect newly

launched attacks or unbalanced datasets. In this study, only a limited number of machine-learning techniques were examined. A comparison of the techniques discussed in this study with other machine-learning techniques that could be used to detect intrusions would be interesting.

2.4.3 Probability-based Learning Techniques

As a statistical method for selecting features, (Hnamte, Balram, et al., 2022) proposes using the dice similarity coefficient as a statistical method. As a result of categorizing the transactions in IoT networks as positive or negative, the proposal is designed to select optimal features to prevent intrusion. By indicating whether the network infrastructure transaction can be accessed, the proposed model significantly improves the classification performance of the network infrastructure transaction. IoT-Sentinel is used to evaluate the proposed model output. A naive Bayes classifier is used for scaling the proposed model output. Due to the need to evaluate the proposed IDS from the perspective of scalability, robustness, and overhead, this study has a significant disadvantage. Furthermore, it assumes that the training data will be independent and equally distributed, which is unrealistic in IoT networks that contain dynamic sources with heterogeneous distributions. (Onah, Abdullahi, Hassan, Al-Ghusham, et al., 2021) has proposed GANBADM as part of the security research of fog computing environments. Feature selection algorithms are Genetic Algorithm Wrapper, and Naive Bayes classifiers are utilized in the model. Since only one dataset was used in this study, the results may not apply to other datasets because this model was constructed using a dataset known as the NSL-KDD, which achieved 99.73% accuracy.

2.4.4 Error-based Learning Techniques

This method utilizes Deep Convolutional Neural Networks, which use deep learning techniques, to detect IoT device attacks, developed by (Baniasadi, Rostami, Martín, & Kaveh, 2022) using (NSBPSO-based DCNN) algorithm to detect IoT device attacks. This project also includes the development of a neighborhood search-based particle swarm optimization algorithm (NS-BPSO) for enhanced exploration and exploitation capabilities. During the experiment, UNSW-NB15 and Bot-IoT trained the proposed model using the proposed algorithm's advantages to improve accuracy and performance. On the test dataset, NSBPSO-DCNN achieved a performance of 99.41%, whereas, on the training dataset, it achieved a performance of 98.86%. As this study is computationally expensive, it may not be feasible for real-time intrusion detection due to IoT systems' lack of much-labeled data. Researchers developed several deep learning and machine learning algorithms integrated with IDS to detect abnormal network traffic patterns. (Naveed et al., 2022) proposed a new method to detect IDS since there is still a low detection rate. A two-stage implementation process is involved in implementing this model. The first stage will involve selecting features, including Chi-squares, ANOVA, and PCA. Based on a 99.73 percent accuracy rate, a deep neural network model was also developed using the NSL-KDD dataset for classifying traffic into normal and abnormal modes. Therefore, the framework could have performed better on other datasets in this study, representing a significant limitation. A data imbalance creates a difficult challenge since average data are outnumbered by attack data, resulting in incorrect classification of attack data as the decision border may deviate. In addition to creating a difficult challenge, a data imbalance can result in incorrect classification of attack data as the decision border may deviate when attack data outnumber average data. (Ponmalar & Dhanakoti, 2022) propose an intrusion detection system that addresses the inherent complexities asso-

ciated with big data, the complexities associated with heterogeneous forms of security data, and the inherent complexities associated with big data environments. As a result of the suggested model, nine new assault types were identified in the UNSW-NB15 dataset with 96.29% accuracy, increasing classification accuracy as well. Network traffic is analyzed using SVM, and features are selected using CGO. It is important to note that this study has several limitations, including that a large amount of labeled data must be obtained, which is expensive and difficult to obtain. The researcher (Ullah & Mahmoud, 2021) developed a new IDS model by combining the RFE feature selection approach and CNN algorithms. As part of the assessment of the accuracy of the model, the MQTT-IoT-IDS2020 dataset, which is comprised of three layers of CNNs and multiclass classifications, is used. A CNN was the only deep-learning model considered in this study, although other models, including RNNs, LSTMs, and GRUs, may be more suitable for IoT networks.

2.4.5 Hybrid Model

A new model for IDS has been developed by (Hnamte & Hussain, 2023) using RFE feature selection and CNN algorithms. Three layers of CNNs and multiclass classification using CNN algorithms are utilized. The model's accuracy is determined by evaluating it on the MQTT-IoT-IDS2020 dataset. In this study, CNNs were the only deep learning model considered, and no other deep learning models were used. These strata are referred to as D1 and D2. In a hybrid Deep CNN-BiLSTM model, the researchers used CNNs to extract distinct features and Bidirectional LSTM models to capture long-term and short-term relationships. The proposed model was trained on the CICIDS2018 and Edge IIoT datasets. The accuracy rates on the CICIDS2018 dataset were 100 percent with and without GPUs. In Edge IIoT datasets with a GPU, accuracy rates are 99.46 percent, and loss rates are

0.0080; without a GPU, accuracy rates are 99.62 percent and loss rates are 0.0081. Compared to typical deep-learning models, the suggested model requires considerable training time. There are limitations to this study due to the computationally expensive costs associated with training and deploying this approach, as well as the quality of the training data, which can be used to detect a variety of attacks. As the sophistication and number of attacks increase, anomaly-based NIDS may become more difficult to handle large amounts of data, such as RNNs, LSTMs, and GRUs, which may serve as an alternative for IoT networks. The research conducted by (Talukder et al., 2023) consists of integrating deep learning and machine learning into a system to increase the detection rate while maintaining reliability. This research tested and evaluated KNN, RF, DT, CNN, ANN, MLP, machine learning, and deep learning algorithms. Based on the results of the research, it was determined that the RF model was the most accurate based on the SMOT algorithm's balance of data and XGBoost. The proposed model should have been developed considering environmental factors such as network traffic and bandwidth. The proposed model is based on KDDCUP '99, which has 99 percent accuracy, and CIC-MalMem-2022, which has 100 percent accuracy. This hybrid NIDS model was developed by (Balyan et al., 2022) and utilizes improved random forest techniques (IRF) in conjunction with enhanced genetic algorithms and particle swarm optimization (EGA-PSO) techniques. EGA-PSO was used to improve minor data and create a balanced dataset in order to obtain more precise insight into the properties of small samples. The approach reduces dimensions, identifies key features, enhances the GA using a multi-objective function, selects the best feature, and enhances fitness outcomes by enhancing the vector, enhancing the GA, enhancing the GA, and selecting the best feature. This reduces false positives and true positives and enhances the effectiveness of analyzing key features. As part of iterative training, less critical features are removed, several

decision trees are incorporated, and the classification accuracy of the classifier is monitored in order to avoid overfitting in a decision-response framework. The HNIDS approach achieves 98.97% accuracy in BCC and 88.14% accuracy in MCC, significantly better than SVM, RF, LR, NB, and LDA based on experimental results on NSL-KDD datasets. The study measured only four parameters: accuracy, TPR, precision, and FPR. Other parameters were not considered, including CPU utilization and time. Deep learning techniques have been utilized in ImmuneNet (Akshay Kumaar et al., 2022) to recognize attacks. With multiple feature engineering processes, oversampling methods, and hyperparameter optimization methods, the proposed framework achieves high performance and accuracy. Data from CIC IDS 2017, CIC IDS 2018, and CIC bell DNS 2021 have been used to compare the proposed model to XGB, Random Forest, Decision Trees, and Logistic Regression. According to the CIC Bell DNS 2021 dataset, ImmuneNet achieved 99.19% accuracy. A continually updated system is necessary to protect patient records from future cyberattacks. As a result, the model cannot learn independently and, therefore, can take considerable time and effort to learn recent attacks. Using RF, SVM, and DT, as well as an ensemble voting classifier suggested by (Patil et al., 2022), an IDS was produced with 96.25% accuracy on the CICIDS-2017 dataset, demonstrating the importance of trust in achieving human-machine interaction success. Using a Learnable Machine Interface makes it possible to describe predictions in a way that is easily understood, which facilitates selecting models, assessing trust, improving unreliable models, and providing insight to experts and non-experts alike. A single dataset was utilized to evaluate performance in this study, which may not reflect real-world scenarios or data distributions. Furthermore, it does not address privacy, accountability, or trust issues when it comes to intrusion detection. It examines (Rincy N & Gupta, 2021) various machine learning techniques that can be used to analyze different types

of assaults. By combining high-accuracy machine learning algorithms with low false positive rates, CAPPER selects the best feature subsets for hybrid NID-Shield NIDS. CAPPER utilizes the optimal feature subsets from the CFS in conjunction with Wrappers for selecting feature subsets, where the CFS provides a subset of features free from redundant and unnecessary components. It is possible to obtain accurate and high-quality feature sets for training and testing by combining filtering and wrapping approaches. Wrapper algorithms enable the production of highly accurate feature subsets for training and testing using induction learning algorithms, and wrapping algorithms result in accurate and high-quality features. With the UNSW NB15 and NSL-KDD20% datasets, a hybrid CAPPER algorithm has been tested with a performance accuracy of 99.89%, while a performance accuracy of 99.90% was achieved in both cases. In order to determine whether the proposed model reflects the latest attack trends, it was not evaluated against a recent dataset. The proposed model is based on Naive Bays, random forests, and J48 (C4.5). A hybrid anomaly detection system has been proposed by (Dutta, Choraś, Kozik, & Pawlicki, 2021) by combining Deep Neural Networks with Classical Auto Encoders (CAE). It is first necessary to initialize the input data using a CAE encoder before testing the DNN classification model against the UNSW-NB15 benchmark dataset. The second stage evaluates the proposed model against the benchmark dataset. According to the findings of this study, the proposed model has an accuracy rate of 91.29 percent, which is significantly higher than those of previously benchmarked methods. However, the study was based on only a single dataset, which should have been more recent. Due to the rapid growth of IoT environments and their increasing threats, existing machine-learning algorithms will need help to identify new attacks. In (Liu, Gu, & Wang, 2021) study, a hybrid anomaly detection system was proposed by combining Deep Neural Networks with Classical Auto Encoders (CAE). Before evaluating the pro-

posed classification model against the UNSW-NB15 benchmark dataset, it is necessary to initialize the input data using a CAE encoder. The proposed classifier is evaluated against the benchmark dataset during the second stage. Following the results of this study, the proposed model has an accuracy rate of 91.29 percent, a significantly higher accuracy rate than previous benchmarked models. The study, however, utilized only a single dataset, which should have been more recent. Due to the rapid growth of IoT environments and their increasing threats, existing machine-learning algorithms will not be capable of identifying new threats due to the rapid growth of IoT environments. It has been proposed by (Aldallal & Alisa, 2021) to combine hybrid IDS techniques with GA and SVM machine learning techniques in order to secure data in cloud environments. The hybrid IDS method uses the GA to classify network data into normal and abnormal behavior. Then, the GA is employed to select features for the hybrid IDS method. The KDD CUP 99 dataset was used in this study. A folded dataset is used to train and test the system to determine its capability with such data. During the study, only one dataset was used. The accuracy level in the study was 99.3%. Therefore, the proposed model may not be accurate when applied to other datasets and may not reflect the most recent trends in attacks.

2.4.6 SDN Model

Recently, Software Defined Networking (SDN) has gained popularity as a way to overcome traditional system limitations by controlling the network through one sophisticated piece of software. Due to the separation of control and data planes, the SDN system will simplify network management and improve programmability. Furthermore, the recommended strategy would provide SDN protection that is more precise, efficient, and scalable than current techniques, thereby overcoming the limitations of current techniques. Many businesses use network detection algorithms to identify and organize

harmful traffic. However, it is difficult to identify these algorithms in unbalanced datasets (Elsayed, Hamada, Abdalla, & Elsaid, 2023). In order to detect network threats in the SDN-IoT environment, an IDS model-based LSTM algorithm with four hidden layers was developed by (Chaganti, Suliman, Ravi, & Dua, 2023). Through this method, t-SNE-based learning is displayed that has been acquired by the DL models over various hidden layers. LSTM models have been trained on the SDN-IoT dataset and have achieved a 97.1% accuracy rate. For LSTM models to detect and classify network attacks in real-time, large amounts of labeled data are required. As part of the SDN attack detection and identification techniques, methods have been proposed by (Alzahrani & Alenazi, 2021) for detecting and identifying attacks. Attack detection was demonstrated using the following machine learning models: DT, RF, and XGBoost. Testing was conducted on the NSL-KDD dataset, achieving 95 percent accuracy. This study used just one dataset, which requires updating and may include current attacks. Using SDN-enabled deep learning-driven approaches, (Javeed, Gao, Khan, & Ahmad, 2021) propose a highly effective, scalable, and cost-effective solution to threat detection within IoT environments. Using Cu-BLSTM and Cu-DNNGRU classifiers, this hybrid model was trained and tested using the CICIDS2018 dataset, achieving 99.87% accuracy. Due to the limited data used in the study, it is still being determined how well the proposed model will perform on other datasets, especially those with different characteristics. In an SDN environment, IDS utilize six ML algorithms for detecting and mitigating MR-DDoS attacks: RT, REP Tree, SVM, J48, and MLP. It utilizes a flexible modular architecture proposed by (Perez-Diaz, Valdovinos, Choo, & Zhu, 2020). MLP has a 95% accuracy rate. This study does not evaluate the performance of other datasets for testing and evaluation, as other datasets may have different distributions or attack patterns. As part of the testing and evaluation process, (CIC) DoS datasets were used.

2.5 Discussion

The four challenges of IDS are low detection rate, false alarm rate, response time, and unbalanced datasets. It is due to some limitations in both the data source method and the detection method that these challenges arise. These challenges can be addressed by three detection methods: machine learning, data mining, and statistical techniques. Machine learning will be used as a detection technique in this study. According to the advantages and disadvantages of the machine learning family, error-based learning is the most effective method with IDS. From related work, it can be concluded that a hybrid model has the highest accuracy result, as illustrated in (Hnamte & Hussain, 2023), (Talukder et al., 2023), where they achieved 100% accuracy. However, when probability-based learning is used, the accuracy result is less, as shown in in (Hnamte et al., 2022); the proposed model's accuracy is 89.7%. The accuracy of information-based learning and error-based learning is always over 98%. Additionally, feature selection will significantly impact the accuracy result, as demonstrated by the research proposed by (Onah et al., 2021), where the accuracy result of the proposed model is 99.73%. Furthermore, the feature selection method can have a significant effect on the accuracy of the ML model, as shown in the following related papers: (Hnamte & Hussain, 2023), (Talukder et al., 2023), (Balyan et al., 2022), (Akshay Kumaar et al., 2022), (Patil et al., 2022), (Rincy N & Gupta, 2021), (Dutta et al., 2021) each falls under the hybrid model approach, but they differ in accuracy. Both feature selection and parameter selection can significantly impact the model's accuracy. In all cases, SDN shows an accuracy of 95% or greater. The literature review considers no specific network architecture for applying ML-based IDS. However, this study considers the network architecture the principal component for applying ML-based IDS. Also, the related works provided one-layer security and did not consider the organizations' requirements if they were under primetime or special

events. Most related studies test one ML model with multiple datasets or one dataset trained with multiple models, so there needs to be more explicit about whether the results are accurate. Moreover, most of the mentioned related works are unrelated to specific domains, such as cloud computing or the Internet of Things, so this field has been left empty. Table 2.2 is the summary of related work. Consequently, a novel centralized network architecture based on multi-model IDS is proposed, called multi-model IDS based on traffic load. Further, the following datasets will be used in this study: CIC-MalMem-2022, a recent dataset; CIC-IDS-2018 and CIC-IDS-2017 datasets, the most commonly used related studies. In addition to this, the CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017 datasets serve as valuable resources for researchers involved in intrusion detection and malware analysis. As a result of their real-world representation, diversity of attack types, labeled data, public accessibility, and status as benchmarking standards, they are widely used to develop and evaluate new security solutions. Finally, several learning approaches will be used for this experiment, including hybrid models, information-based learning techniques, and error-based learning techniques. As they achieve high levels of accuracy. The following are the learning approaches and algorithms that will be used in the experiments:

- Hybrid models: Stacking Classifier and Voting Classifier enable different estimators to predict the result.
- Information-based learning techniques: AdaBoost is a powerful and versatile meta-algorithm particularly useful for cybersecurity applications as it can handle imbalanced and noisy datasets (Alharbi & Alsubhi, 2021). Random Forest: most used in related studies, and XGBoost: is one of the most powerful gradient-boosting algorithms available (Wang et al., 2021). Generally, tree-based algorithms such as RF, Adaboost, and XGbbost have demonstrated high efficiency and

effectiveness in handling binary and categorical data and identifying complex feature relationships. Various ML applications use these algorithms, such as classification, regression, and anomaly detection.

- Error-based learning techniques: Support Vector Machine: Researchers commonly use SVM algorithms because of their versatility, robustness, and capability to handle complex data. They are particularly good at handling classification problems but can also be used for regression and anomaly detection.

Table 2.2: Summary of LR

Ref.	Domain	Method	Algorithms	Dataset	Feature Selection	Accuracy
(Attou et al., 2023)	CC	Information	RF	BoT-IoT	Graphic Visualization	99.99%
(Hnamte & Hussain, 2023)	-	Hybrid	CNN, BiLSTMs, DNN	CICIDS 2018	CNN	100%
(Chaganti et al., 2023)	SDN-IoT	SDN	LSTM	SDN-IoT	t-SNE	97.1%
(Talukder et al., 2023)	-	Hybrid	RF	CIC-MalMem-2022	XGBoost	100%
(Saheed et al., 2022)	IoT	Information	XgBoost	UNSW-NB15	PCA	99.99%
(Baniasadi et al., 2022)	IoT	Error	DCNN	Bot-IoT	NSBPSO	98.86%
(Balyan et al., 2022)	-	Hybrid	IRF	NSL-KDD	EGA-PSO	99.97%
(Naveed et al., 2022)	Network Traffic	Error	DNN	NSL-KDD	Chi square, ANOVA, and PCA	99.73%
(Ding et al., 2022)	-	Similarity	KNN	CIC-IDS-2017	-	95.86%
(Hnamte et al., 2022)	IoT	Probability	Naïve Bayes	IoT Sentinel	DSC	89.7%
(Ponmalar & Dhanakoti, 2022)	Big data	Error	SVM	UNSW-NB15	CGO	96.29%
(Akshay Kumar et al., 2022)	-	Hybrid	ImmuneNet	CIC Bell DNS 2021	All feature selection	99.19%

Ref.	Domain	Method	Algorithms	Dataset	Feature Selection	Accuracy
(Patil et al., 2022)	–	Hybrid	Voting	CIC-IDS-2017	Correlation heatmap technique	96.25%.
(Rincy N & Gupta, 2021)	Network	Hybrid	Naive Bayes ,RF, J48 (C 4.5) ,CFS	NSL-KDD20%	CAPPER	99.90%
(Dini & Saponara, 2021)	LAN	Similarity	KNN	LAN traffic used by the US Air Force	MDS	99.57%
(Dutta et al., 2021)	-	Hybrid	DNN	UNSW-NB15	CAE	99.979%
(Ullah & Mahmoud, 2021)	IoT	Error	CNN	MQTT-IoT-IDS2020	RFE	99.99%
(Alzahrani & Alenazi, 2021)	SDN	SDN	XGBoost	NSL-KDD	–	95.95%
(Javeed et al., 2021)	SDN -IoT	SDN	Cu-BLSTM and Cu-DNNGRU	CIC-IDS-2018	–	99.87%.
(Liu et al., 2021)	-	Hybrid	k-means ,RF ,LSTM, and CNN	CIC-IDS-2017	–	99.91%
(Aldallal & Alisa, 2021)	CC	Hybrid	SVM	KDD CUP 99	GA	99.3%
(Onah et al., 2021)	Fog Computing	Probability	Naïve Bayes	NSL-KDD	GA-Wrappe	99.73%
(Seth et al., 2021)	IoT	Information	LightGBM	CIC-IDS-2018	RF&PCA	97.73%
(Perez-Diaz et al., 2020)	SDN	SDN	MLP ₃₄	(CIC) DoS	-	95%

2.6 Summary

This chapter summarizes IDS and its methods and techniques and discusses its advantages and disadvantages. Moreover, it discusses the main IDS challenges as well as detection methods. Machine learning is a detection method that can address these challenges, as it is one of the detection methods available for solving IDS problems. Moreover, the ML algorithms are categorized according to the ML family, and the advantages and disadvantages of each category are covered. The chapter also discusses related work. Network architectures and models are proposed based on the discussion of related work.

Chapter 3

Methodology

3.1 Overview

This chapter will discuss a centralized network architecture for an organization that uses IDSs based on ML. Also, it will cover the proposed IDSs based on the ML model.

3.2 Multi-model IDS based on traffic load

Network architectures with a centralized architecture can significantly impact the performance of intrusion detection systems that utilize machine learning algorithms. By implementing a centralized architecture, security threats can be detected more accurately due to improved visibility and control. The centralized control plane can be utilized by IDSs based on ML algorithms to collect and analyze network data from various devices, thereby detecting anomalous behavior and potential threats (Shetabi & Akbari, 2020).

In addition, centralized architectures allow for uniform enforcement of security policies across the network, reducing the risk of misconfigurations or inconsistencies that may result in false positive alerts from the IDS. By defining and distributing security policies across all network devices, the central controller ensures consistent protection against cyber threats (Valdovinos, Pérez-Díaz, Choo, & Botero, 2021). Moreover, security incidents or breaches can be responded to more rapidly if the network is centralized. By isolating compromised devices, implementing remediation measures, and identifying the affected network areas, the central controller can quickly identify the affected network areas. By centralized control, security incidents can be responded to more efficiently and efficiently, minimizing their impact (Valdovinos et al., 2021). In contrast to decentralized network architectures, multiple controllers can present additional challenges to managing and coordinating. Keeping ML-based IDS performing and effective can be challenging when it comes to ensuring consistent policy enforcement and efficient communications between controllers and network devices (Ahmad & Mir, 2021). Furthermore, decentralized architectures may result in additional overhead associated with communication and coordination between controllers, potentially impeding the performance of ML-based IDS (Khan et al., 2020). Decentralized and centralized network architectures are distinct approaches to managing computer systems, networks, and data storage. According to the centralized network architecture, these networks have the following key differences: Hierarchical structure, which is controlled and managed by a central authority; a single point of failure, which can lead to data breaches, data outages, and censorship; data is stored in a central location, resulting in vulnerabilities and privacy concerns; as the central authority struggled to manage the growing amount of data and users, scalability and performance may be limited (Chen, Wu, & Christofides, 2021). With a decentralized network architecture, the flat structure, where no single en-

tity controls the entire network, increases security and privacy since data is stored in multiple nodes, increasing its resilience to attacks and censorship and improving its scalability and performance, as it can adapt to increased numbers of users and data storage, as well as lowering reliance on a central authority, which can improve the network's overall resilience (Chen et al., 2021). Therefore, centralized and decentralized network architectures can benefit from ML-based IDS, but the choice will depend on the network's specific requirements and constraints. In contrast, decentralized architectures provide better privacy and resilience, whereas centralized architectures may provide enhanced security and control (Helmy & Nayak, 2020), (Chen et al., 2021). A summary of the key differences between centralized and decentralized network architectures can be found in Table 3.1, and a summary of the advantages and disadvantages of centralized and decentralized network architectures can be found in Table 3.2.

Table 3.1: A comparison of the critical differences between centralized and decentralized network architectures

Feature	Centralized Network	Decentralized Network
Control	One central authority	Multiple nodes are involved in the distribution
Processing power	A centralized server or cloud service provider	Multiple nodes are involved in the distribution
Data storage	A centralized server or cloud service provider	Multiple nodes are involved in the distribution
Communication	The central hub facilitates the flow of data and requests	Each node communicates with the others and collaborates with them
Single point of failure	Yes	No
Resilience to attacks	Low	High
Resilience to failures	Low	High
Scalability	High	Limited
Complexity	Simple	Complex
User autonomy	User autonomy	High

Table 3.2: Comparing the advantages and disadvantages of centralized and decentralized network architectures

Network Type	Advantages	Disadvantages
Centralized Network	It is easy to manage. Allocation of resources in an efficient manner. Providing strong security.	Having a single point of failure. A reduction in resilience. The reduction of user autonomy.
Decentralized Network	Adaptability to failure. A reduction in censorship. Enhancement of user autonomy.	An increase in complexity. Consensus mechanisms that are difficult to achieve. Scalability is limited.

As shown in Figure 3.1, a centralized network architecture is proposed that utilizes a multi-model IDS based on traffic load. Network traffic com-

ing into the organization from WAN will first be inspected by the firewalls, which can block the IPs that contain malicious traffic; this can be accomplished by configuring the IPS to block traffic from malicious IP addresses by adding these addresses manually to the IPS's blocklist. An IPS is a comprehensive security solution that not only detects but also actively prevents various forms of cyber threats by combining the functionality of both IDS and firewalls. Before network traffic is distributed to the sub-organizations, global inspection will be done first through multi-model IDS, so there are two models for ML-based IDS. Each IDS model will work according to the traffic entering the network. To calculate traffic in the network, routers and switches are configured to collect and report network traffic statistics. The amount and type of traffic passing through these devices may help determine the volume and type of traffic in the network. The amount of traffic is the threshold for choosing the best model; the network administrator assigns the threshold value. As long as traffic is higher than the threshold value, the fastest model will be active; for example, the network traffic is higher than the threshold value in the organization in prime time or on specific occasions. If network traffic is lower than the threshold value, the model with the highest accuracy will be active after choosing the best model to inspect and detect the traffic. Then, the traffic will be distributed to sub-organizations. The IDS, placed in these sub-organizations, is based on a model with high accuracy in detecting malicious activity. Algorithm 1 and Algorithm 2 describe how the proposed architecture works. A multi-model IDS based on traffic load can also be applied to SDN, although suitability will depend on several factors, including network complexity, hardware requirements, and application requirements (Urrea & Benítez, 2021).

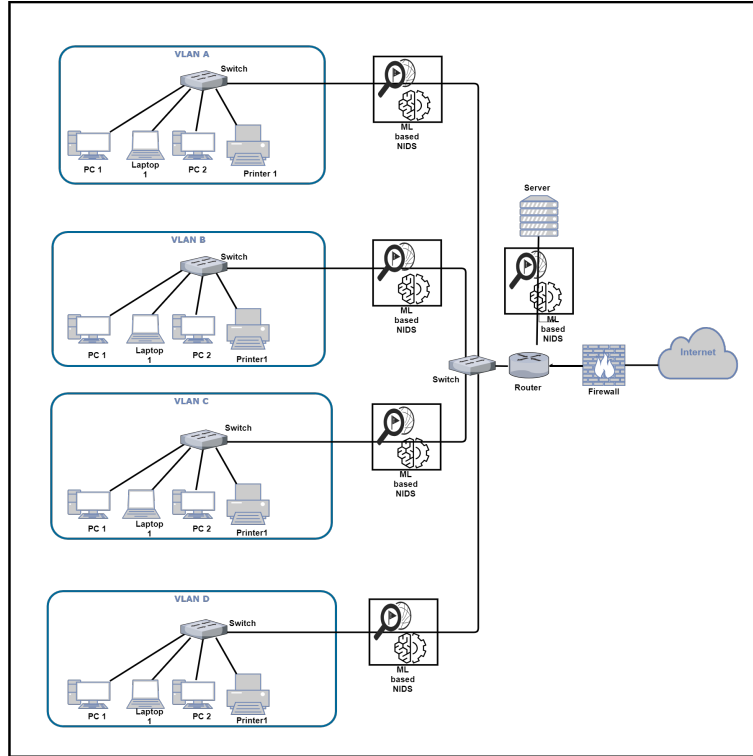


Figure 3.1: Multi-model based network traffic load

The steps of Algorithm 1 are as follows: First, the psutil module is imported, providing an interface to the operating system's processes and information. The following function, `get_network_traffic()`, returns the current network traffic. As a first step, it performs the `psutil.net_io_counters(pernic=True)` method returns a dictionary of network interfaces indicating how many bytes were received and sent on each interface. After iterating over the dictionary of interfaces, the function adds the `bytes_recv` and `bytes_sent` attributes for each interface to the `network_traffic` variable. Finally, the function returns the `network_traffic` variable. The final line of code calls the function `get_network_traffic()`, which prints the current network traffic to the console. According to algorithm 2, the first line imports the pickle module, which is used for serializing and deserializing Python objects. Following these two lines, `model1`, which is the fastest to detect the malicious, and `model2`, which has high accuracy, are loaded from .pkl files created by training the

models on data of features and labels. The data and the threshold are two arguments in the `select_model()` function. Pandas DataFrames contain the features of the new data point wish to predict. The threshold is the amount of network traffic used to determine which model to use. The threshold value is controlled by the network admin. In order to determine the current network traffic, the function first uses the `get_network_traffic()` function, as described in Algorithm 1. When the network traffic exceeds the threshold, the function uses the `model1`. Alternatively, the function makes the prediction using `model2`. The function then returns the result. In the final two lines of code, the test data is read from a CSV file, and the `select_model()` function is called to make the prediction. The prediction is then printed to the console.

Algorithm 1 To get network traffic

```

1: Import psutil
2: function GET_NETWORK_TRAFFIC
3:   interfaces ← psutil.net_io_counters(pernic=True)
4:   network_traffic ← 0
5:   for all interface ∈ interfaces do
6:     network_traffic ← network_traffic + interfaces[interface].bytes_recv + inter-
       faces[interface].bytes_sent
7:   end for
8:   Return network_traffic
9:   network_traffic ← get_network_traffic()
10:  Print the current network traffic is network_traffic bytes.

```

Algorithm 2 To select the best model

```
1: Import pandas as pd
2: Import pickle
3: Load the model1 into the variable model1
4: Load the model2 into the variable model2
5: function SELECT_MODEL
6:     data  $\leftarrow$  sample
7:     threathold  $\leftarrow$  threathold
8:     traffic  $\leftarrow$  get_network_traffic()
9:     if traffic > threathold then
10:         prediction  $\leftarrow$  model1.predict(data)
11:     else
12:         prediction  $\leftarrow$  model2.predict(data)
13:     end if
14:     Return prediction
15: sample  $\leftarrow$  pd.read_csv('testFeatures.csv')
16: prediction  $\leftarrow$  select_model(sample, threathold=1000000)
17: Print the prediction is prediction
```

3.3 Benchmark Dataset

This section provides information about the datasets used in this study to assess the effectiveness of the methodology proposed. The CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017 datasets serve as valuable resources for researchers involved in intrusion detection and malware analysis. As a result of their real-world representation, diversity of attack types, labeled data, public accessibility, and status as benchmarking standards, they are widely used to develop and evaluate new security solutions. A variety of attacks are present in the datasets, including probe attacks, such as port scans, ping sweeps, and network scans; U2R attacks, such as buffer overflows and SQL injections; and 2L attacks, such as remote login attacks, as well as Trojan horse attacks (Louk & Tama, 2022), (Qazi, Faheem, & Zia,

2023), (Hassan, Abdullahi, Aliyu, Yusuf, & Abdulrahim, 2022). In Table 3.3, the datasets and attack classes are described in detail. The architecture of the model is illustrated in Figure 3.2.

3.3.1 CIC-MalMem-2022

A renowned malware dataset was used to simulate a memory-based detection algorithm using CIC-MalMem-2022, an obfuscated malware dataset. In order to provide a realistic simulation, renowned malware was used to create it. The dataset contains both malicious and benign memory dumps. Malware that has been obfuscated can conceal its identity to avoid detection. Additionally, CIC-MalMem-2022 contains 56 features that are used in machine learning algorithms. The dataset has various types of attacks, including U2R attacks, R2L attacks, probe attacks, malware attacks, and botnet attacks (Louk & Tama, 2022).

3.3.2 CIC-IDS-2018

This dataset contains two patterns: benign and seven intrusion techniques available to the public. The dataset includes 80 statistical features, including packet length, volume, and bytes, all measured forward and reverse. The raw data is updated as of today and has been collected multiple times. Researchers can currently access the dataset online. These include several types of attacks, such as DDOS attacks, Brute-force DoS attacks, Brute-force SSH attacks, Infiltration attacks, Heartbleed attacks, Botnet attacks, and Web attacks. In detail, the dataset contains Probe attacks, such as port scans, ping sweeps, and network scans. It also contains U2R attacks, such as buffer overflows and SQL injection attacks. Additionally, it contains 2L attacks, such as remote login attacks and Trojan horse attacks (Qazi et al., 2023).

3.3.3 CIC-IDS-2017

A network traffic flow analysis with CICFlowMeter is a combination of traditional network attacks and real-world network intrusions that have been created by the CIC in 2017. The analysis uses 83 features and 28,30,743 instances of genuine and different types of attacks using timestamps, destination IP addresses, sources, protocols, and attacks. Also included in the dataset are several common attack scenarios, including brute force attacks, infiltration, port scanning, web attacks, bots, and denial of service attacks. The dataset consists of Probe attacks, such as port scans, ping sweeps, and network scans. Additionally, the dataset consists of U2R attacks, such as buffer overflows and SQL injection attacks, as well as 2L attacks, such as remote login attacks and Trojan horse attacks (Hassan et al., 2022).

Table 3.3: A comparison of the CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017 datasets

Datasets	The number of sample	Classes
CIC-MalMem- 2022	58,596	Benign,Trojan Horse,Ransomware,and Spyware .
CIC-IDS-2018	16,232,943	Benign, Botnet,DDOS, Brute-force Dos, In-filtration, Web attack, Brute-force SSH, and Heartbleed
CIC-IDS-2017	2,830,743	Benign, Infiltration, Port scanning, Denial of Service (DoS), web attacks, brute force, and bots

Some of the main differences and similarities between them are as follows: Memory dumps in CIC-MalMem-2022 can be used to detect obfuscated malware. 58,596 records of benign and malicious memory dumps of

spyware, ransomware, Trojan horses, and other malware families are available on the CIC website since 2022. CIC-IDS-2018 is a continuation of CICIDS-2017 in generating realistic network traffic based on profiles. More attack scenarios have been added to the CIC-IDS-2018 dataset, including brute force, web attacks, botnet attacks, and DDoS attacks. CIC has published the dataset on its website since it was created in 2018. The database contains 16,545,685 records of network packets, accompanied by 80 unique characteristics, such as source and destination IP addresses, ports, protocols, and payloads. Also, the CICIDS-2017 dataset precedes the CIC-IDS-2018 dataset, in which profiles simulate both normal and malicious network traffic. This dataset has four scenarios: infiltration, DoS, port scans, and heart-bleed. The dataset was created in 2017 and can be accessed through the CIC website. It contains 2,830,743 records of network packets with 79 features, similar to those found in CIC-IDS-2018 (Louk & Tama, 2022), (Qazi et al., 2023), (Hassan et al., 2022). These three datasets share several similarities:

- These resources are publicly available and free to use for research purposes.
- Each label indicates what type of attack it represents, such as ransomware or DDoS, or what type of traffic it represents, such as benign or malicious.
- These methods are suitable for applying machine learning techniques, such as supervised and unsupervised learning, for anomaly-based detection and classification.

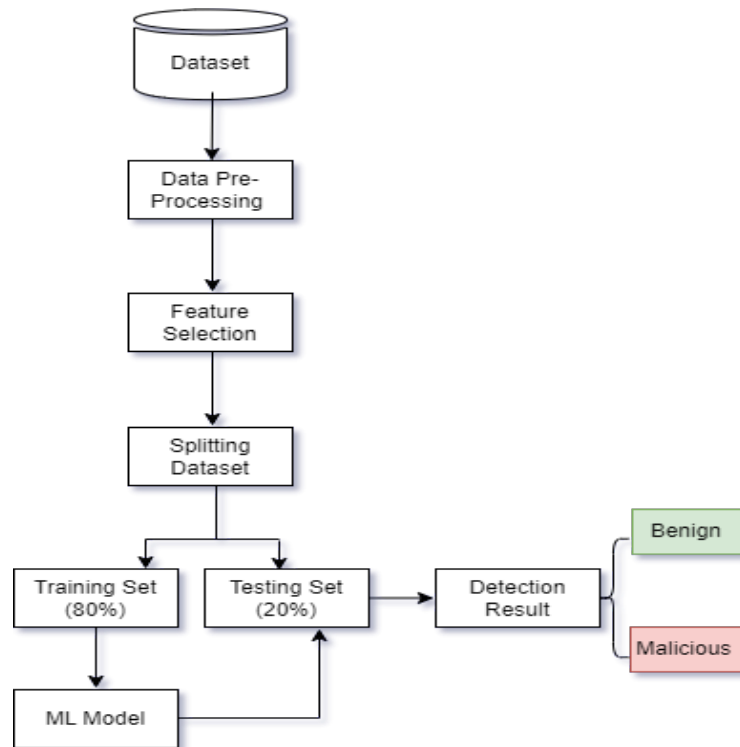


Figure 3.2: Machine learning model architecture

3.4 Features Selection

Machine learning experimentation requires an evaluation of the suitability of various feature selection techniques for different datasets within the field. In machine learning, feature selection is integral in identifying the most relevant and informative features from a given dataset, an essential step in the process. Choosing a subset of relevant features allows us to improve the performance of ML models, improve their interpretability, and reduce the computational complexity of their outputs (Pudjihartono, Fadason, Kempa-Liehr, & O’Sullivan, 2022). Selecting feature selection techniques is based on the dataset’s characteristics and the specific ML task being conducted. The effectiveness of each feature selection technique depends on the type of data, the distribution of features, and whether there is noise or irrelevant information in the data (Pudjihartono et al., 2022). There are two types of

feature selection in this study. For CIC-MalMem-2022 and CIC-IDS- 2017 datasets, SelectKbest is applied, while VarianceThreshold is used as feature selection for CIC-IDS-2018. There are several benefits of using SelectKBest to select features for intrusion detection tasks. It is a simple, efficient, and effective technique that can enhance the performance of ML algorithms. However VarianceThreshold method is ideal for complex datasets because it is simple, efficient, and can quickly identify subsets of features with high variance (Izquierdo et al., 2021).

3.4.1 Univariate Feature Selection

The sci-kit-learn library in Python provides feature selection using SelectKBest. It uses a statistical test to determine the top K features. Many types of statistical tests are available, including chi-squared, F-tests, mutual information tests, and others. Selecting the first k features with the highest scores from the input dataset, SelectKBest retains them (Izquierdo et al., 2021). To determine the strength of the relationship between a feature and a response variable, SelectKBest uses a univariate feature selection method. This method offers a greater understanding of data because of its simplicity and ease of use (though it could be optimized for better generalization because of the simplicity and ease of use of these methods). SelectKBest simplifies the modeling process when working with many features by identifying the most essential features to make accurate predictions (Izquierdo et al., 2021).

3.4.2 Removing Features with Low Variance

In the VarianceThreshold algorithm, features with variances that do not meet a certain threshold are removed from the analysis. As a default, all zero-variance features, which have the same value across all samples, are re-

moved. Variance is typically calculated as the squared distance from the mean divided by the average squared value (Brownlee, 2020). The variance of binary data (values 0 and 1) can be calculated using the following formula:

$$Var[X] = p(1 - p) \quad (3.1)$$

Where p indicates the proportion of 1s in the dataset.

3.5 Data Preparation

- **Data Pre-Processing**

1. **Data Cleaning**

There are several data preparation techniques applied to each dataset.

3.5.0.1 CIC-MalMem-2022

Firstly, the category column has been removed from the dataset since it contains the same data as the Class column. The data in both columns consists of benign and malware data. Also removed 534 duplicate rows from the dataset. During the run of the model, the error message 'sklearn/feature selection_univariate_selection.py:112:userwarning feature[3 9 50 54] are constant' was received, so the following columns containing constant values were removed: (pslist.nprocs64bit, handles.nport, svcscan.interactive_process_services, and callbacks.ngeneric). The dataset is divided into two categories, namely target and features. The target contains the Class column, and features contain the rest of the dataset columns.

3.5.0.2 CIC-IDS-2018

The dataset contains ten separate CSV files. Nine are for malicious data, and the tenth is for benign data. By using the concatenation function, the ten CSV files become one. In addition, only 5000 records are selected from each CSV file because it is a very large dataset. Afterward, the label column is converted into a category that indicates malicious or benign behavior. In the next step, the following columns are removed from the dataset:

- (a) `dst_ip`, `src_port`, `src_ip`, and `flow_id`: 45,000 records with NaN values in these columns must be dropped.
- (b) Duplicate values are dropped.

Features Engineering:

- (a) The data type for the timestamp column is converted to the `DateTime` type, and three new columns, namely `day`, `month`, and `year`, are added to the dataset. In these three new columns, the value from the timestamp column is taken, and then the timestamp column is dropped.
- (b) Splitting the dataset into features and the target, the data types of all feature columns that are objects are converted to numeric. Sets the values of the features to 0 if the value is infinite to run the code faster and so not affect the accuracy.

3.5.0.3 CIC-IDS-2017

The dataset contains eight separate CSV files, each containing benign and malicious records. The eight CSV files can be concatenated into one using the concatenation function. Moreover, only 50000 records are selected from each CSV file because it

is a huge dataset. After this step, the `_label` column is converted into a category Malicious or Benign. Covert the data types of features to float , because some features are numeric but the data type is object. Sets the values of the features to 0 if the value is infinite. And also replace NaN value with zero. The following columns are removed from the dataset:

- (a) (`_bwd_psh_flags`, `_bwd_urg_flags`, `fwd_avg_bytes_bulk`, `_fwd_avg_packets_bulk`, `_fwd_avg_bulk_rate`, `_bwd_avg_bytes/bulk`, `_bwd_avg_packets/bulk`, `bwd_avg_bulk_rate` , `fwd_urg_flags`, `_cwe_flag_count`, `_rst_flag_count`, `_ece_flag_count`) because they contain constant values.
- (b) A total of 13835 duplicate records have been dropped.

This thesis uses a subset of CIC-IDS-2018 and CIC-IDS-2017 datasets for several reasons: first, manageability: smaller datasets are more manageable, especially when computational resources are limited. Secondly, by concentrating on specific aspects of the data relevant to their study, a subset allows researchers to reduce irrelevant information by focusing on relevant data. The use of subsets is helpful for the preliminary testing of hypotheses and algorithms, ensuring more controlled and manageable experiments. Data quality is also enhanced when a smaller subset is cleaned and preprocessed to ensure high data accuracy. Additionally, subsets can assist in maintaining privacy and adhering to legal restrictions, especially if sensitive information is involved. As a final point, resource efficiency: analyzing a smaller dataset can save time and resources in terms of computational power and labor (Mussmann, Fang, Schmidt, & Jamieson, 2022).

2. Random Undersampling or Oversampling

The CIC-MalMem-2022 dataset and the CIC-IDS-2017 dataset do not require oversampling or undersampling, as shown by Figures 3.3 and 3.5, in contrast to the CIC-IDS-2018 dataset, where the malicious class is greater than the benign class, so the RandomUnderSampler is used, as shown in Figure 3.4.

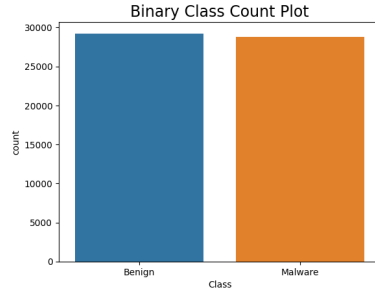


Figure 3.3: Binary class CIC-MalMem-2022 dataset

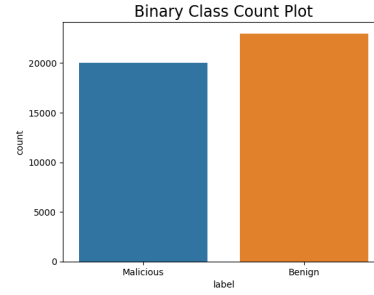


Figure 3.4: Binary class for CIC-IDS-2018 dataset

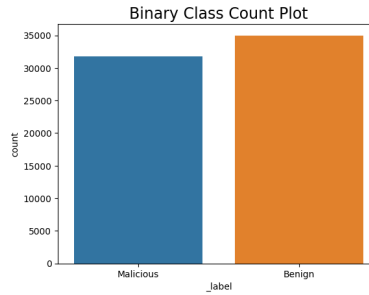


Figure 3.5: Binary class for CIC-IDS-2017 dataset

3. **LabelEncoder for Target** Using the LabelEncoder function from sklearn in Python to encode the categorical data, the value of the Class column in the CIC-MalMem-2022 dataset and the values in the label column in CIC-IDS-2018, CIC-IDS-2017 dataset converts into 0 for Benign and 1 for Malware.
4. **Data normalization** Using StandardScaler function from sklearn in Python to scale the dataset's features. Categorical Features : To handle the missing categorical data, the following are used,

FunctionTransformer class and get_dummies() is a pandas function. DateTime Features: The (FunctionTransformer(np.int32(x))) is applied to convert from float to integer.

- **Feature Selection** The feature selection in the CIC-MalMem-2022 dataset and the CIC-IDS-2017 dataset are selected using SelectKbest, where K is 25 in the CIC-MalMem-2022 dataset, as shown in Table 3.4, and K is 30 in the CIC-IDS-2017 dataset, as shown in Table 3.5. There are several benefits of using SelectKBest to select features for intrusion detection tasks. It is a simple, efficient, and effective technique that can enhance the performance of ML algorithms. Since the CIC-IDS-2018 dataset is complex, using VarianceThreshold to select features is preferable. It is an excellent choice for complex datasets because it is simple, efficient, and can quickly identify high-variance subsets of features. All features with low variance are excluded from the CIC-IDS-2018 dataset. The threshold value for this experiment is 0. Table 3.6 displays some features for the CIC-IDS-2018 dataset.

Table 3.4: Features for CIC-MalMem-2022 dataset.

Index	Feature Name
0	pslist.nppid
1	pslist.avg_threads
2	pslist.avg_handlers
3	dlllist.ndlls
4	dlllist.avg_dlls_per_proc
5	handles.nhandles
6	handles.nevent
7	handles.ndesktop
8	handles.nkey
9	handles.nthread
10	handles.ndirectory
11	handles.nsemaphore
12	handles.ntimer
13	handles.nsection
14	handles.nmutan
15	ldrmodules.not_in_load
16	ldrmodules.not_in_init
17	ldrmodules.not_in_mem
18	ldrmodules.not_in_load_avg
19	ldrmodules.not_in_mem_avg
20	svcsan.nservices
21	svcsan.kernel_drivers
22	svcsan.process_services
23	svcsan.shared_process_services
24	svcsan.nactiv

Table 3.5: Features for CIC-IDS-2017 dataset

Index	Feature Name
0	_destination_port
1	_fwd_packet_length_max
2	_fwd_packet_length_min
3	_fwd_packet_length_mean
4	_fwd_packet_length_std
5	bwd_packet_length_max
6	_bwd_packet_length_min
7	_bwd_packet_length_mean
8	_bwd_packet_length_std
9	_flow_packets/s
10	_flow_iat_std
11	_flow_iat_max
12	_fwd_iat_std
13	_fwd_iat_max
14	bwd_iat_total
15	fwd_packets/s
16	_min_packet_length
17	_max_packet_length
18	_packet_length_mean
19	_packet_length_std
20	_packet_length_variance
21	_psh_flag_coun
22	_ack_flag_count
23	_urg_flag_count
24	_average_packet_size
25	_avg_fwd_segment_size
26	_avg_bwd_segment_size
27	idle_mean
28	_idle_std
29	_idle_max

Table 3.6: Some of Features for CIC-IDS-2018 dataset.

Index	Feature Name
0	dst_port
1	protocol
2	flow_duration
3	tot_fwd_pkts
4	tot_bwd_pkts
5	totlen_fwd_pkts
6	totlen_bwd_pkts
7	fwd_pkt_len_max
8	fwd_pkt_len_mean
9	day
10	month

- **Splitting the Datasets** By using `train_test_split` function, the data(features and target) split into training data with value 80% and test data with value 20%.

The datasets were significantly reduced in size from their original dimensions due to applying data preprocessing and feature selection methods. Consequently, most CIC-MalMem-2022 data needed more relevant, redundant, and accurate. Moreover, the CIC-MalMem-2022 dataset was reduced to 25, indicating that the remaining features are most important for predictive modeling. Despite reducing new samples to 32713, a significant portion of the CIC-IDS-2018 dataset required more variety, relevance, and accuracy. Additionally, the number of features was reduced to 72, indicating that the remaining features were the most important for predictive modeling. According to the decrease in the number of new samples to 66757, a significant portion of the CIC-IDS-2017 dataset needed to be more varied, relevant, and accurate. Moreover, the number of features was reduced to 30, indicating that the remaining features were most relevant to the predictive modeling

process. Table 3.7 summarizes the number of features and samples for each dataset. As a result of this reduction in data size and feature count, several benefits can be obtained, including:

- As a result of focusing on the most relevant and reliable data, machine learning algorithms can identify patterns and relationships within the data more accurately, resulting in more accurate predictions (Kelleher et al., 2020).
- Minimized computational complexity: Smaller datasets require fewer computational resources, making training and executing machine learning models more efficient (Y. Zhang et al., 2020).
- The interpretation of the results is enhanced when fewer features are used as fewer features make it easier to identify the underlying factors that influence the study's results (Linardatos, Papastefanopoulos, & Kotsiantis, 2020).
- By reducing the dimensionality of a model, overfitting can be mitigated by eliminating features which do not contribute significantly to predictive power. By reducing the number of features contributing little to prediction, dimension reduction techniques can assist in mitigating overfitting (Kelleher et al., 2020).

Table 3.7: A comparison of the CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017 datasets after preprocessing steps

Datasets	The number of samples	The number of features
CIC-MalMem- 2022	58062	25
CIC-IDS-2018	32713	72
CIC-IDS-2017	66757	30

3.6 Machine Learning Models

This experiment will use different learning approaches to train and test the datasets. Including hybrid models (Voting, Stacking) because they enable different estimators to predict the result; information-based (AdaBoost, Random Forest, and XGBoost.), the tree-based algorithms have demonstrated excellent efficiency and effectiveness in analyzing binary and categorical data and identifying complex feature relationships. In addition to classification, regression, and anomaly detection, these algorithms are used for various ML applications, and error-based learning techniques(SVM); because of their versatility, robustness, and capability to handle complex data, researchers often use SVM algorithms in classification, regression, and anomaly detection problems. The mentioned models were trained and tested separately for the following datasets: CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017.

3.6.1 AdaBoost

In this experiment, the AdaBoost classifier is set with an attribute `n_estimators = 50`, which is the number of weak learners to be trained iteratively. The weak learner in this experiment is the Decision Tree Classifier, a default estimator with a default `max_depth` of 1. Furthermore, the estimator was set to `None`, and the `learning_rate` was set to `default=1.0`.

3.6.2 Voting Classifier

This experiment involves training a Voting classifier using a set of individual estimators and combining the predictions made by these individual estimators into a final prediction. Each estimator independently makes a prediction using this method, and a majority vote or weighted vote is used to determine

the final prediction (Ubale & Gaikwad, 2022). Several classifiers were used in this experiment: AdaBoost Classifier, Logistic Regression, KNeighbors Classifier, Gradient Boosting Classifier, Random Forest Classifier, and Decision Tree Classifier. Also, the following parameter, voting, is assigned as hard and weights as None.

3.6.3 Stacking Classifier

In a stacking classifier, many individual classifiers are combined hierarchically to produce a final prediction. Training several base classifiers on the training data is the first step, and their predictions serve as inputs for a higher-level classifier known as the meta-classifier. Upon training the meta-classifier with the output from the base classifiers, the final prediction is made using the output from the meta-classifier (R. Zhao, Mu, Zou, & Wen, 2022). This experiment utilized the following estimators: AdaBoost Classifier, Logistic Regression, KNeighbors Classifier, Gradient Boosting Classifier, Random Forest Classifier, and Decision Tree Classifier. For the final estimator, the Bagging Classifier was used. Also, the `stack_method` parameter is assigned to the default parameter 'auto'.

3.6.4 Support Vector Machine

An algorithm that finds the parameters with the most significant distance from all multidimensional samples, or feature vectors, is implemented using a hyperplane that is maximally distant from all the samples. New data points can be classified by whether they fall on one or more sides after generating the hyperplane. The regularization parameter refers to the strength of regularization that is inversely proportional to C (Xuan, Hu, Wang, & Liu, 2021). This parameter must be strictly positive, and the value of one was used in this experiment. Additionally, the kernel parameter is assigned

as 'rbf'.

3.6.5 Random Forest

The RF algorithm works by combining multiple decision trees to make predictions. The `n_estimators` are set to default values of 100, and the Gini coefficient is used to examine the quality of the split. Also, `max_depth` parameter is assigned None, so, nodes are expanded until all leaves are pure or all leaves contain fewer samples than `min_samples_split`. The `min_samples_split` in this experiment are assigned to the default value 2.

3.6.6 XGBoost

The algorithm is a type of gradient boosting algorithm whose performance is highly scalable and efficient, and it is commonly used for large-scale data analysis applications. As a result of its ability to handle both sparse and dense data formats, XGBoost is particularly well suited to problems with large numbers of features (Leevy, Hancock, Zuech, & Khoshgoftaar, 2020). It has several parameters; in this experiment, `n_estimators` are used with a default value of 100 trees. Moreover, `max_depth` and `min_child_weight` used the default value 6 and 1 consecutively.

3.7 Performance Evaluation

This section discusses three different approaches to evaluating machine learning models.

3.7.1 Confusion Matrix

Evaluating the performance of the ML algorithms using confusion metrics can provide significant insight into the performance and bias of different algorithms. Table 3.8 presents the Confusion Matrix.

Table 3.8: A confusion matrix for binary classifications

Class/Prediction	Benign	Malicious
Benign	TN	FP
Malicious	FN	TP

In order to determine the accuracy of the model or the correct rate, it is necessary to determine how accurate the model is for the given classification task.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.2)$$

where,

1. The TN denotes the correct prediction for the benign class.
2. The TP indicates the correct prediction for the malicious class.
3. The FP implies an incorrect prediction for the benign class. It is predicted as malicious, but it is actually benign.
4. The FN represents an incorrect prediction for the malicious class. It is predicted as benign, but in reality, it is malicious.

According to the definition of precision, it is the ratio of true positives to all positives.

$$Precision = \frac{TP}{TP + FP} \quad (3.3)$$

Recall can be defined as the ratio between the true positives and the sum of the false negatives plus the true positives.

$$Recall = \frac{TP}{TP + FN} \quad (3.4)$$

An F1 score indicates the degree of similarity between a predicted and true set based on the balance between precision and recall.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \quad (3.5)$$

3.7.2 Learning curve

The learning curve describes how the performance of ML model on a task changes over time or with increasing experience or training. As a result of more data or experience, the model's performance improves. Learning curves can be used to diagnose problems with models, such as over-fitting or under-fitting, and to determine how much training data is required to meet a specific performance requirement (Mohr & van Rijn, 2022). Machine learning uses learning curves to illustrate how a model's performance correlates with training experience. A typical learning curve depicts the relationship between performance (often expressed in terms of several training examples) and experience (often expressed in the number of training examples). On the x-axis, experience is plotted. Various performance metrics are used to measure performance, including accuracy and error rate, and plotted on the y-axis (Mohr & van Rijn, 2022). Some of the most important characteristics can be summarized in the following (Géron, 2022):

1. **Training Curve:** The training curve shows how the model performs on the training dataset as more training data are presented. Generally, as the model receives more training data, its performance on the training dataset will improve.

2. Validation curves illustrate a model's performance over various validation datasets. These curves can plateau or even worsen after a certain level, indicating that the model has become overfitted.
3. Overfitting and underfitting:
 - In terms of training and validation, an underfitted model is characterized by high error rates since the model is too simplistic to capture the underlying patterns in the data.
 - A model that has been overfitted learns that there is noise in the training data. While the training error decreases, the validation error increases.
4. For the model to generalize well to new data sets, it is ideal for both the training and validation curves to converge at a low error rate.

3.7.3 Hyper-parameter in the Model

Machine learning models perform better when selecting hyper-parameters appropriately because they affect how the algorithm fits the model to the data. For the model to perform better in terms of its complexity or learning speed, it is essential to select appropriate hyper-parameters. In order to maximize a model's performance, it is essential to tune the hyper-parameters (Hamida, El Gannour, Cherradi, Ouajji, & Raihani, 2020). During the creation of machine learning models, hyperparameters are used, settings that are not learned from data and are set before the training process begins (Yu & Zhu, 2020). The following are critical components of hyperparameters (Yu & Zhu, 2020):

1. The following types of hyperparameters can be identified:

- Characteristics specific to a particular model may include the depth of DTs or the number of hidden layers within a neural network, for example.
 - Training-specific factors include learning rate or batch size in gradient descent.
2. A hyperparameter tuning approach involves finding the best parameters to optimize a model's performance on a given task. It can be carried out by grid search, Bayesian optimization, or random search.
 3. In terms of importance:
 - Optimizing hyperparameters can significantly improve a model's performance.
 - It is possible to control overfitting and underfitting by changing hyperparameters such as the regularization parameter.
 4. The challenge of tuning hyperparameters can be computationally expensive and time-consuming, especially for models with a large number of complex interactions between parameters.

A powerful tool within the sci-kit-learn library is the GridSearchCV tool, which can be used to perform hyperparameter tuning in machine learning efficiently and effectively. A grid of possible combinations is created by systematic evaluation of hyperparameters. The best combinations are evaluated using cross-validation and selected based on their performance. This method can automate hyperparameter tuning, increasing model performance and avoiding manual trial and error (Hamida et al., 2020). A description of each hyperparameter and its values is provided in Table 3.9.

Table 3.9: Hyper-parameter and tuning values for each ML algorithm

ML Algorithm	Hyper-parameter	Values
AdaBoost	n_estimators	80,100,150,200 ,and 250
Voting	voting	Hard or Soft
Stacking	passthrough	True or False
SVM	Regularization parameter (C)	0.1, 0.9, 10.0, 50.0, and 100.0
RF	max_depth	2,3,5,10, and 20
XGBoost	Learning rate	0.5,0.7,1,2, and 10

3.8 Summary

A multi-model IDS based on traffic load network architecture is proposed in this chapter, along with ML-based IDS models. The chapter also provides detailed information regarding the preprocessing steps for the three datasets. Further, the chapter provides information about the evaluation process using the confusion matrix, learning curve, and hyperparameters.

Chapter 4

Results and Discussion

4.1 Overview

This experiment was conducted on an HP laptop with an Intel(R) Core(TM) i7-8550U processor @ 1.80GHz 2.00GHz, 16.0 GB RAM, and Windows 11. An intrusion detection system has been developed using machine learning techniques on the open-source Google Colab Notebook platform. This chapter will present and analyze the results.

4.2 Results

4.2.1 AdaBoost

This model was trained on the first dataset, CIC-MalMem-2022, with an accuracy of 99.94%. To evaluate the model, different outputs are evaluated. Based on the Confusion Matrix shown in Figure 4.1, the model cor-

rectly predicted 5805 malicious and 5802 benign instances. However, two malicious instances were incorrectly predicted as benign, and four benign instances were incorrectly predicted as malicious. For CIC-IDS-2018, accuracy was 99.11%, and the confusion matrix in Figure 4.2 indicates that the model correctly predicted 2059 malicious instances and 4418 benign instances. Nevertheless, 13 malicious instances were incorrectly predicted as benign, and 45 benign instances were incorrectly predicted as malicious. CIC-IDS-2017 has an accuracy rate of 98.64%, and a confusion matrix in Figure 4.3 shows that the model accurately predicted 6235 malicious instances and 6936 benign instances. However, 115 malicious instances were incorrectly predicted as benign, while 66 benign instances were incorrectly predicted as malicious. A summary of the results for each dataset is shown in Table 4.1

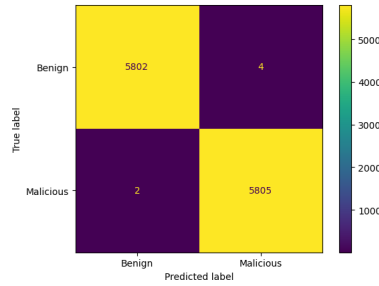


Figure 4.1: Confusion matrix on the CIC-MalMem-2022 dataset for AdaBoost

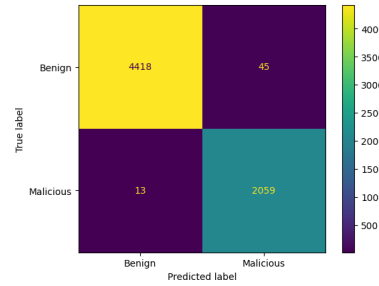


Figure 4.2: Confusion matrix on the CIC-IDS-2018 dataset for AdaBoost

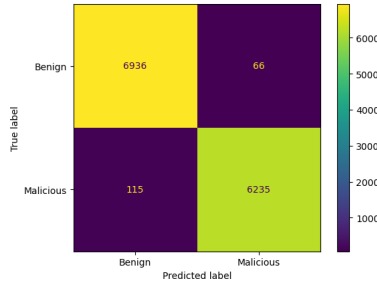


Figure 4.3: Confusion matrix on the CIC-IDS-2017 dataset for AdaBoost

Table 4.1: Summary of Adaboost model results

Dataset	Accuracy	Precision	Recall	f1-score	CPU Us- age	Training Time	Testing Time
CIC-MalMem-2022	99.94%	99.96%	99.93%	99.94%	0.0000054	4.9 s	0.10s
CIC-IDS-2018	99.11%	99.37%	97.86%	98.61%	0.0000074	5.46 s	0.082 s
CIC-IDS-2017	98.64%	98.18%	98.95%	98.56%	0.0000072	8.06 s	0.11 s

4.2.1.1 Learning Curve

The learning curve CIC-MalMem-2022 dataset in Figure 4.4 shows that the training score is 100% accurate in every rotation; however, cross-validation scores show different accuracy levels in each cycle. The cross-validation score starts at 99.82%, reaches 99.88% with training examples ranging from 20000 to 30000, and then continues with an accuracy of 99.86%. These training and validation percentage results indicate no evidence that overfitting has occurred. As shown in Figure 4.5, the learning curve for the CIC-IDS-2018 dataset shows that the training scores have an accuracy of around 99.99% in every cycle, but cross-validation scores range from 94% to 96%. Comparing the training and cross-validation results shows that the model was balanced. The learning curve shown in Figure 4.6 for the CIC-IDS-2017 dataset indicates that the model performs well both in training and cross-validation, as noted in the high training and cross-validation scores. There is a slight difference between the training and cross-validation scores, which is a typical pattern. Consequently, there is no overfitting of the training data to the model.

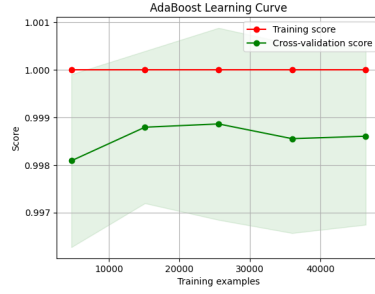


Figure 4.4: Learning Curve on the CIC-MalMem-2022 dataset for AdaBoost

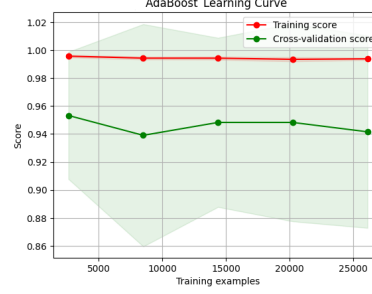


Figure 4.5: Learning Curve on the CIC-IDS-2018 dataset for AdaBoost

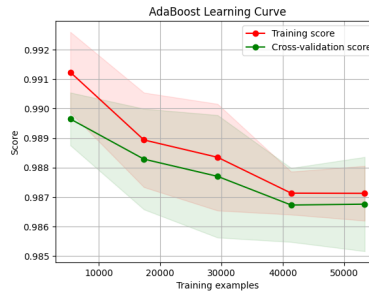


Figure 4.6: Learning Curve on the CIC-IDS-2017 dataset for AdaBoost

4.2.1.2 Hyperparametr in the Model

For the AdaBoost model, $n_estimators$ is an effective parameter. Five values are assigned to check the best $n_estimators$, and they are 80,100,150,200 and 250; for the experiment on the CIC-MalMem-2022 dataset, the best value is 80 with time 0.102 s; moreover, all values give 100% accuracy as shown in Figure 4.7 so, in this case, it is necessary to check the time. Table 4.2 summarizes the results.

Table 4.2: AdaBoost hyperparameter results for CIC-MalMem-2022 dataset

mean_score_time	param_model__n_estimators	mean_test_score
0.102534 s	80	0.999311
0.123210 s	100	0.999311
0.180753 s	150	0.999311
0.235057 s	200	0.999311
0.282698 s	250	0.999311

For the CIC-IDS-2018 dataset, the best parameter is 100, with a mean test score of 99.40% and a mean score time of 0.191 seconds. Figure 4.8 illustrates the relationship between the `n_estimators` and their accuracy. According to Figure 4.9, the best parameter for the CIC-IDS-2017 dataset is 250. The mean test score and mean test time for the previous dataset were 99.24% and 0.358 seconds, respectively.

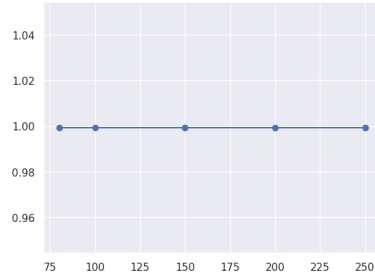


Figure 4.7: Hyperparameter tuning on the CIC-MalMem-2022 dataset for AdaBoost

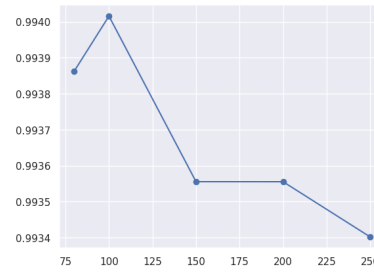


Figure 4.8: Hyperparameter tuning on the CIC-IDS-2018 dataset dataset for AdaBoost

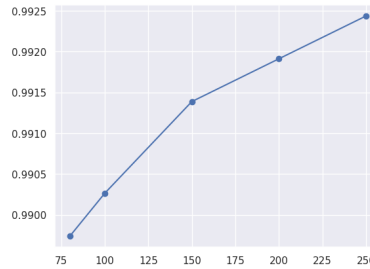


Figure 4.9: Hyperparameter tuning on the CIC-IDS-2017 dataset dataset for AdaBoost

4.2.2 Voting Classifier

Based on the first dataset CIC-MalMem-2022, the model was trained and had an accuracy result of 99.96%. According to the confusion matrix in Figure 4.10, the model correctly predicted 5806 malicious and 5803 benign instances. However, one malicious instance was incorrectly predicted as benign, and three benign instances were incorrectly predicted as malicious. The accuracy result for the CIC-IDS-2018 dataset is 99.58%, and based on Figure 4.11, the model correctly predicted 2077 malicious instances and 4431 benign instances. In addition, 27 benign instances were incorrectly predicted as malicious, while no malicious instances were incorrectly predicted as benign. While an accuracy result for the CIC-IDS-2017 dataset is 99.31% by this model, the confusion matrix in Figure 4.12 indicates that the model correctly predicted 6279 malicious and 6982 benign instances. This model made incorrect predictions regarding 69 malicious instances and 22 benign instances. Different results are considered for model evaluation, as illustrated in Table 4.3.

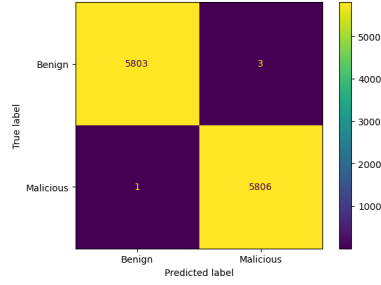


Figure 4.10: Confusion matrix on the CIC-MalMem-2022 for Voting classifier

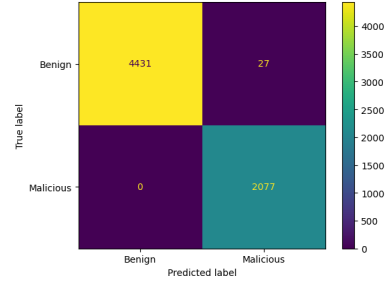


Figure 4.11: Confusion matrix on the CIC-IDS-2018 dataset for Voting classifier

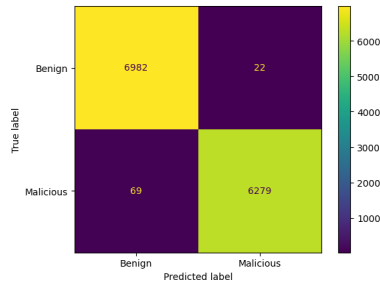


Figure 4.12: Confusion matrix on the CIC-IDS-2017 dataset for Voting classifier

Table 4.3: Summary of Voting classifier model results

Dataset	Accuracy	Precision	Recall	f1-score	CPU Usage	Training Time	Testing Time
CIC-MalMem-2022	99.96%	99.98%	99.94%	99.96%	0.0000055	24.73 s	2.27 s
CIC-IDS-2018	99.58%	100%	98.71%	99.35%	0.0000070	36.99 s	2.20 s
CIC-IDS-2017	99.31%	98.91%	99.65%	99.28%	0.0000075	46.18 s	4.66 s

4.2.2.1 Learning Curve

The learning curve for the CIC-MalMem-2022 dataset in Figure 4.13 indicates that the training score is 100% accurate in each cycle, meaning the model is well-trained to handle the data. In contrast, cross-validation scores are accurate between 99.80% and 99.90%, demonstrating the model's ability to identify and learn new datasets. Compared to the cross-validation score, which ranges from 94% to 96%, the CIC-IDS-2018 datasets' training scores achieved 99.99% accuracy in every cycle, as shown in Figure 4.14. With more examples of data, a classifier learns more about the data, and as the number of training examples increases, the cross-validation score also increases at a slower rate. For the CIC-IDS-2017 datasets, as shown in Figure 4.15, the training score starts at 99.51%, then drops to 99.46%, while the cross-validation score starts at 99.30% and then decreases to about 99.28% in accuracy. As a result of its effectiveness in learning, the voting classifier will likely generalize well to new data.

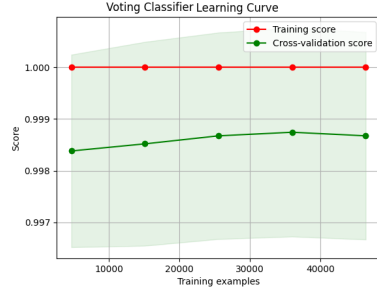


Figure 4.13: Learning curve on the CIC-MalMem-2022 dataset for Voting classifier

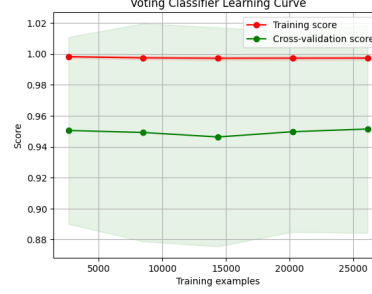


Figure 4.14: Learning curve on the CIC-IDS-2018 dataset for Voting classifier

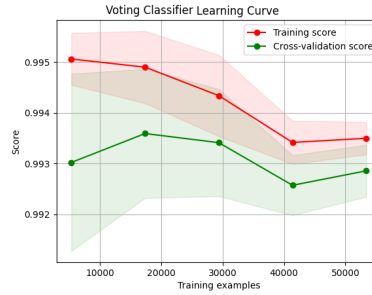


Figure 4.15: Learning curve on the CIC-IDS-2017 dataset for Voting classifier

4.2.2.2 Hyperparametr in the Model

There are two options for voting in the Voting model: hard or soft. The hard option is based on a majority vote, while the soft option is based on the average probability. In the CIC-MalMem-2022 dataset, soft parameters were found to be the most effective, with a mean test score of 99.90% and a mean score time of 0.38 seconds. On the other hand, the hard option results in a mean test score of 99.88%, and the mean test time is 0.76 seconds. Moreover, the best parameter for the CIC-IDS-2018 dataset is also value soft, with a mean score test of 99.44% and a mean score time of 0.22 seconds. Furthermore, the best parameter value for the CIC-IDS-2017 dataset is also soft, with 99.39% mean test scores and 0.78 seconds mean test score times.

Table 4.4 summarizes the results for the three datasets.

Table 4.4: Voting hyperparameter results for CIC-MalMem-2022, CIC-IDS-2018 ,and CIC-IDS-2017 datasets

Datasets	mean_score_time	param_model_voting	mean_test_score
CIC-MalMem-2022	1.222537 s	hard	0.998881
CIC-MalMem-2022	1.086421 s	soft	0.999139
CIC-IDS-2018	0.378114 s	hard	0.994339
CIC-IDS-2018	0.224448 s	soft	0.994492
CIC-IDS-2017	1.634265 s	hard	0.992211
CIC-IDS-2017	0.787651 s	soft	0.993934

4.2.3 Stacking Classifier

In this model, the accuracy result for the CIC-MalMem-2022 dataset is 99.98%. According to the confusion matrix in Figure 4.16, the model correctly predicted 5808 malicious and 5803 benign instances. However, one malicious instance and one benign instance were incorrectly predicted. In the CIC-IDS-2018 dataset, the model achieved a 99.61% accuracy, and per the confusion matrix in Figure 4.17, 2083 malicious instances were correctly predicted, whereas 4427 benign instances were correctly predicted. However, the model predicted four malicious instances as benign and 23 benign instances as malicious. According to the CIC-IDS-2017 dataset, the accuracy rate is 99.86%, and based on the confusion matrix in Figure 4.18, the model predicts only two malicious samples as benign. Different results are considered to evaluate the model, as shown in Table 4.5.

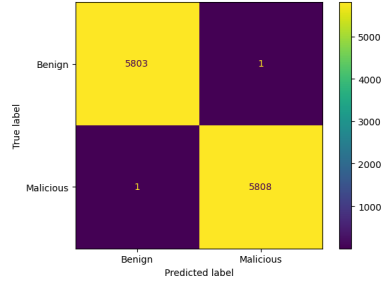


Figure 4.16: Confusion matrix on the CIC-MalMem-2022 for Stacking classifier

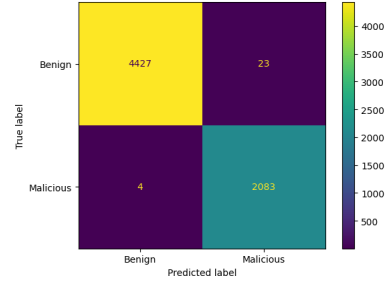


Figure 4.17: Confusion matrix on the CIC-IDS-2018 dataset for Stacking classifier

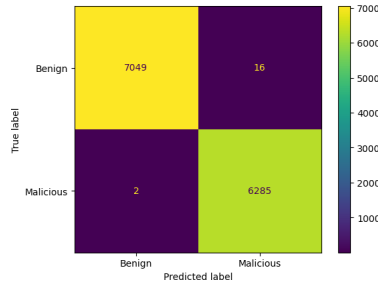


Figure 4.18: Confusion matrix on the CIC-IDS-2017 dataset for Stacking classifier

Table 4.5: Summary of Stacking classifier model results

Dataset	Accuracy	Precision	Recall	f1-score	CPU Usage	Training Time	Testing Time
CIC-MalMem-2022	99.98%	99.98%	99.98%	99.98%	0.0000086	178.92 s	2.90 s
CIC-IDS-2018	99.61%	99.66%	99.13%	99.40%	0.0000095	238.36 s	1.64 s
CIC-IDS-2017	99.86%	99.66%	99.74%	99.85%	0.0000071	184.86 s	2.93 s

4.2.3.1 Learning Curve

According to Figure 4.19, the training score will increase as the number of training examples increases. As the model receives more examples of the data, it becomes more knowledgeable about the data. As a result, the cross-validation score is also increasing, indicating that the model is not overfitting the training data. In this case, a good sign is that the model is learning the data effectively and can also generalize to new data. A stacking learning curve in Figure 4.20 for the CIC-IDS-2018 dataset shows that the training and cross-validation scores will increase as the number of training examples increases. By seeing more examples, the stacking model can better understand the data; the cross-validation score will eventually plateau around 95.51%, whereas the training score will continue to reach 100%. The CIC-IDS-2017 learning curve in Figure 4.21 shows that the training score increases with an increasing number of training examples and cross-validation. A training score of 99.66% is achieved and continues to reach 99.85%; a cross-validation score of 99.25% is achieved and continues to get 99.81%.

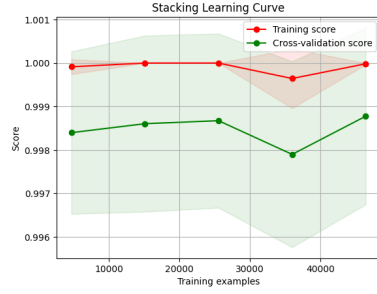


Figure 4.19: Learning curve on the CIC-MalMem-2022 dataset for Stacking classifier

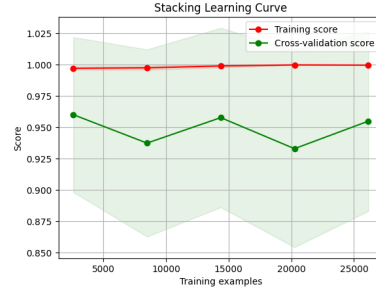


Figure 4.20: Learning curve on the CIC-IDS-2018 dataset for Stacking classifier

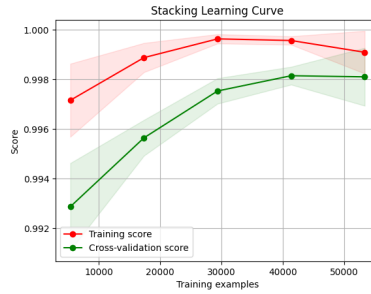


Figure 4.21: Learning curve on the CIC-IDS-2017 dataset for Stacking classifier

4.2.3.2 Hyperparametr in the Model

This model's passthrough parameter can be set to 'True' or 'False'. The 'True' value indicates that the final estimator will be trained on both the predictions of the estimators and the original training data. Accordingly, both the errors made by the estimators, as well as the correct predictions made by the estimators, will be learned by the final estimator. When 'False' is set, the final estimator will only be trained based on predictions made by the estimators, which means that it will only learn from their errors. The final estimator for this experiment is the Bagging Classifier. In the CIC-MalMem-2022 dataset, the best parameter is 'True' with a mean test score is 99.94% with a mean test time of 0.640 seconds. However, the

best parameter for the CIC-IDS-2018 dataset is 'False' with a mean test score of 99.38% and a mean score time of 0.45 seconds. However, the best parameter value for the CIC-IDS-2017 dataset is 'True' with a mean test score of 99.30% and a mean score time of 0.414534 seconds. A summary of the results of the hyperparameter analysis is shown in Table 4.6

Table 4.6: Stacking classifier hyperparameter results for CIC-MalMem-2022, CIC-IDS-2018, CIC-IDS-2017 datasets

Datasets	mean_score_time	param_model_ passthrough	mean_test_score
CIC-MalMem-2022	0.277158 s	True	0.999311
CIC-MalMem-2022	0.280968 s	False	0.999225
CIC-IDS-2018	0.390066 s	True	0.993727
CIC-IDS-2018	0.459702 s	False	0.993880
CIC-IDS-2017	0.414534 s	True	0.993035
CIC-IDS-2017	0.354623 s	False	0.992660

4.2.4 SVM

This model achieves a 99.93% accuracy rate for the CIC-MalMem-2022 dataset. According to Figure 4.22, the model correctly predicted 5808 malicious instances and 5798 benign instances. However, one malicious instance was incorrectly predicted as benign, and six benign instances were incorrectly predicted as malicious. For the CIC-IDS-2018 dataset, the accuracy result is 97.41%. Based on Figure 4.23, the confusion matrix shows that the model correctly predicted 1976 malicious and 4390 benign instances. However, 41 malicious instances were incorrectly predicted to be benign, and 128 benign instances was incorrectly predicted as malicious. In the CIC-IDS-2017 dataset, accuracy was achieved at 94.04% by this model, and the confusion matrix in Figure 4.24 indicates that the model incorrectly

predicted 741 malicious samples as benign, and 54 benign samples as malicious. Table 4.7 outlines different criteria for evaluating the model.

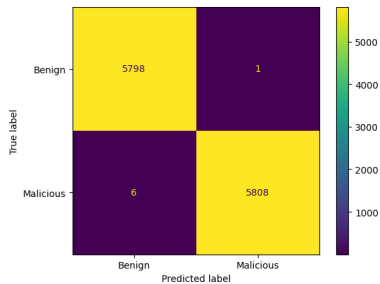


Figure 4.22: Confusion matrix on the CIC-MalMem-2022 dataset for SVM

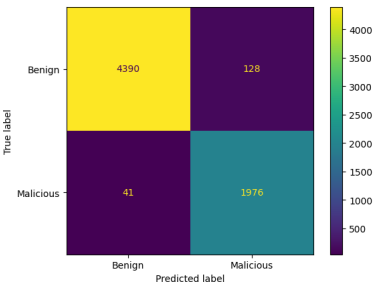


Figure 4.23: Confusion matrix on the CIC-IDS-2018 dataset for SVM

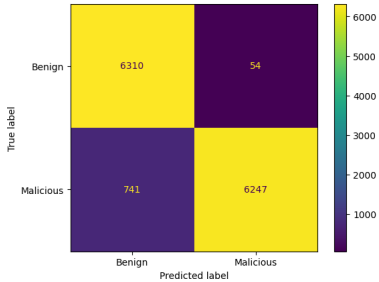


Figure 4.24: Confusion matrix on the CIC-IDS-2017 dataset for SVM

Table 4.7: Summary of SVM model results

Dataset	Accuracy	Precision	Recall	f1-score	CPU Usage	Training Time	Testing Time
CIC-MalMem-2022	99.93%	99.89%	99.98%	99.93%	0.0000084	1.59 s	0.44 s
CIC-IDS-2018	97.41%	97.96%	93.91%	95.89%	0.0000081	7.02 s	1.56 s
CIC-IDS-2017	94.04%	89.39%	99.14%	94.01%	0.0000051	36.14 s	9.81 s

4.2.4.1 Learning Curve

For the CIC-MalMem-2022 dataset, the learning curve, as shown in Figure 4.25, indicates that as the number of training examples increases, the training score rises. The training score starts with 99.88%, a little higher than the cross-validation score, which begins with 99.84% accuracy; however, the cross-validation score decreases after 40000 samples. As shown in Figure 4.26, the training and cross-validation scores for the CIC-IDS-2018 dataset increase with increasing training examples. Nevertheless, the cross-validation score becomes plateaued after a certain point, whereas the training score continues to increase. The learning curve for the CIC-IDS-2017 dataset in Figure 4.27 indicates that the training score increases with the number of training examples because the classifier becomes more proficient at analyzing the data as more samples are viewed. However, cross-validation scores plateau after a certain number of examples. This results in the classifier needing help to learn about the data, and the additional training examples do not improve the classifier's performance.

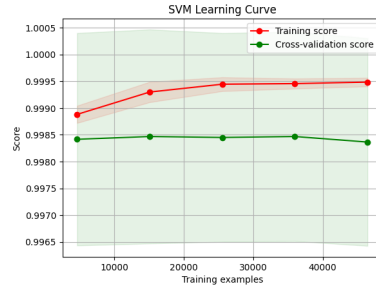


Figure 4.25: Learning curve on the CIC-MalMem-2022 dataset for SVM

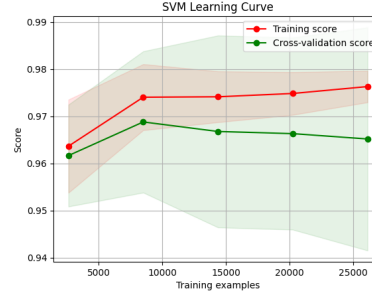


Figure 4.26: Learning curve on the CIC-IDS-2018 dataset for SVM

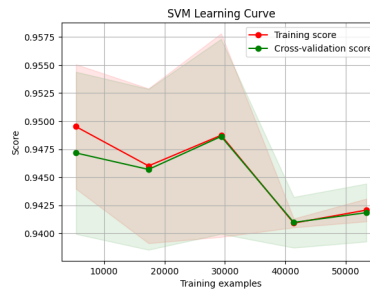


Figure 4.27: Learning curve on the CIC-IDS-2017 dataset for SVM

4.2.4.2 Hyperparametr in the Model

For the SVM model, the Regularization parameter (C) is a very important parameter. As the value of the parameter increases, the classification accuracy improves (or the regression error is reduced); however, overfitting may also occur. In this model, five different values have been assigned to C : 0.1, 0.9, 10.0, 50.0, and 100.0. As shown in Figure 4.28, the best parameter was 50.0 for the CIC-MalMem-2022 dataset. The Figure displays the C values on the x-axis and the accuracy results on the y-axis. Furthermore, the best parameter value for the CIC-IDS-2018 dataset is 100, with mean test scores of 98.31%, as shown in Figure 4.29, and the mean score time is 0.34 seconds. Also, the best parameter for the CIC-IDS-2017 dataset is 100, as shown in Figure 4.30. The mean score for this dataset is 97.20 percent, and

the mean score time is 1.17 seconds.

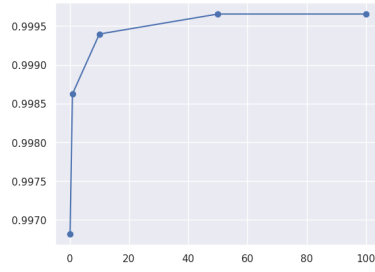


Figure 4.28: Hyperparameter tuning on the CIC-MalMem-2022 dataset for SVM

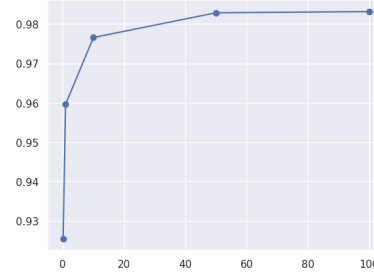


Figure 4.29: Hyperparameter tuning on the CIC-IDS-2018 dataset for SVM

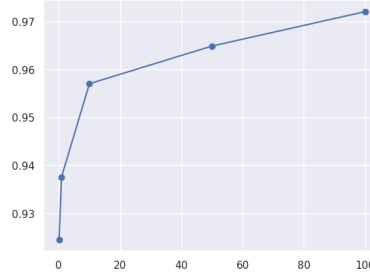


Figure 4.30: Hyperparameter tuning on the CIC-IDS-2017 dataset for SVM

4.2.5 Random Forest

For the CIC-MalMem-2022 dataset, the accuracy result is 99.98%. In Figure 4.31, the model correctly predicted 5808 malicious and 5803 benign instances. However, one malicious instance was incorrectly predicted as benign, and one benign instance was incorrectly predicted as malicious. Using the CIC-IDS-2018 dataset, the accuracy result has been 99.64%, with the confusion matrix shown in Figure 4.32. The model correctly predicts 2071 malicious and 4431 benign instances. Two malicious instances were incorrectly identified as benign, and 24 benign instances were incorrectly identified as malicious. CIC-IDS-2017 achieved 99.40% accuracy; the con-

fusion matrix in Figure 4.33 shows that the model correctly predicted 6287 malicious and 6985 benign instances. However, the model wrongly predicted 66 malicious samples as benign and 14 benign samples as malicious. Table 4.8 details the criteria used to evaluate this model.

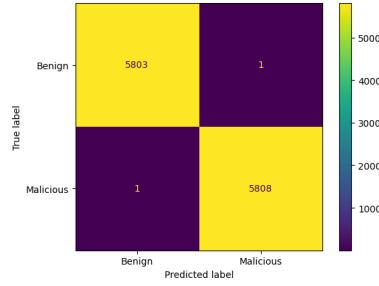


Figure 4.31: Confusion matrix on the CIC-MalMem-2022 dataset for RF

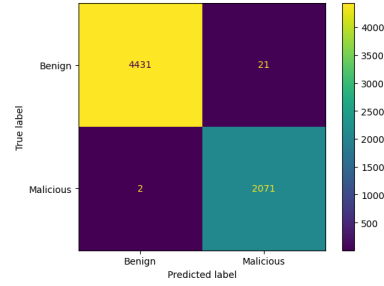


Figure 4.32: Confusion matrix on the CIC-IDS-2018 dataset for RF

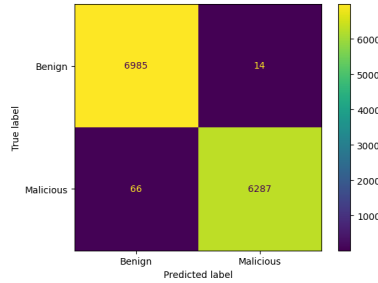


Figure 4.33: Confusion matrix on the CIC-IDS-2017 dataset for RF

Table 4.8: Summary of Random Forest model results

Dataset	Accuracy	Precision	Recall	f1-score	CPU Us- age	Training Time	Testing Time
CIC-MalMem-2022	99.98%	99.98%	99.98%	99.98%	0.000010	6.28 s	0.11 s
CIC-IDS-2018	99.64%	99.90%	98.99%	99.44%	0.000008	6.03 s	0.10 s
CIC-IDS-2017	99.40%	98.96%	98.96%	99.36%	0.0000070	9.92 s	0.18 s

4.2.5.1 Learning Curve

The learning curve for CIC-MalMem-2022 in Figure 4.34 shows that the training score is close to 100%, which indicates that the forest classifier can learn the training data effectively. The cross-validation score, however, is slightly lower than the training score. Figure 4.35 illustrates that the random forest model is learning the CIC-IDS-2018 dataset effectively since each cycle achieved 100% for learning; however, cross-validation shows a slightly lower percentage, starting at around 98.40% and declining to 96.1%. As shown in Figure 4.36, the CIC-IDS-2017 dataset's learning curve indicates the training score begins at 99.71% accuracy and decreases to 99.52. However, the cross-validation score is slightly lower than the training score. It starts at 99.39% and increases to 99.45%.

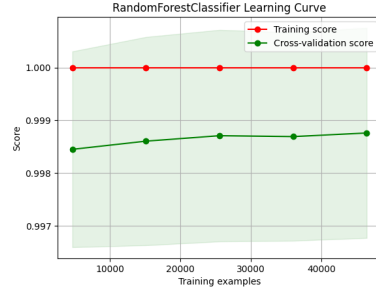


Figure 4.34: Learning curve on the CIC-MalMem-2022 dataset for RF

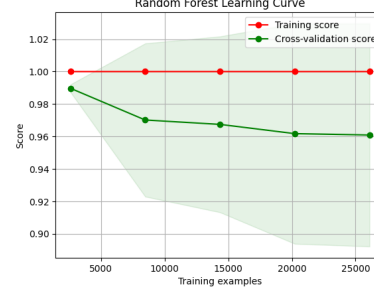


Figure 4.35: Learning curve on the CIC-IDS-2018 dataset for RF

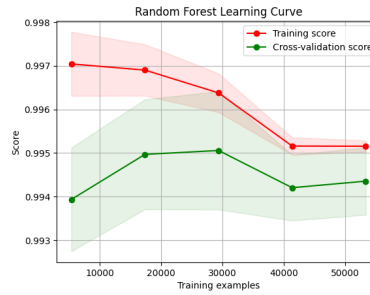


Figure 4.36: Learning curve on the CIC-IDS-2017 dataset for RF

4.2.5.2 Hyperparametr in the Model

The maximum depth parameter in the Random Forest (RF) algorithm is responsible for controlling the depth of each individual tree inside the forest. The depth of a tree is determined by the number of divisions it experiences before reaching a leaf node that lacks any child nodes. In order to address the problem of overfitting, it is possible to use the maximum depth parameter. Setting the maximum depth of the tree too high might potentially cause the tree to excessively memorize the training data, hence resulting in overfitting. The concept of overfitting pertains to the occurrence in which a model exhibits exceptional performance on the training data but fails to demonstrate comparable effectiveness when applied to new, unknown data. When the decision tree is constrained by a low maximum depth, its capacity to

acquire knowledge from the training data is restricted. As a result, this will ultimately result in the tree's incapacity to provide accurate predictions. In the current model, the parameter is assigned the values of 2, 3, 5, 10, and 20. The horizontal axis of the graph corresponds to the greatest depth, whilst the vertical axis is indicative of the mean test findings. The examination of test scores has indicated that the best parameter for the CIC-MalMem-2022 dataset is 20. The mean test score attained is 99.90%, as seen in Figure 4.37. Furthermore, the mean duration required to get these scores is 0.059 seconds. The analysis of Figure 4.38 reveals that the CIC-IDS-2018 dataset attains its best performance when the parameter value is set to 10. The selection of this criterion is correlated with an average examination score of 99.20% and an average time taken to score of 0.056 seconds. Moreover, the best Moreover, drawing upon the evidence shown in Figure 4.39, one may deduce that the most favorable value for CIC-IDS-2017 is 20. The aforementioned conclusion is substantiated by the average test score of 99.34% and the average score time of 0.248 seconds.

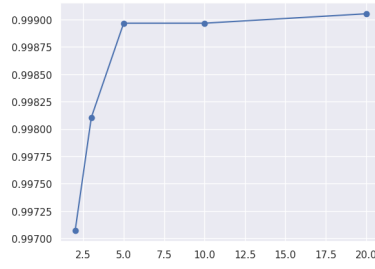


Figure 4.37: Hyperparameter tuning on the CIC-MalMem-2022 dataset for RF

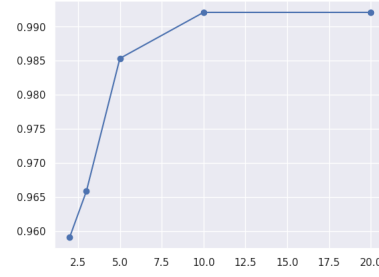


Figure 4.38: Hyperparameter tuning on the CIC-IDS-2018 dataset for RF

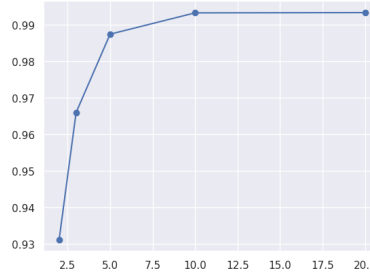


Figure 4.39: Hyperparameter tuning on the CIC-IDS-2017 dataset for RF

4.2.6 XGBoost

For the CIC-MalMem-2022 dataset, this model achieves an accuracy rate of 99.96%. According to Figure 4.40, the model correctly predicted 5806 malicious and 5803 benign instances. However, one malicious instance was incorrectly predicted as benign, and three benign instances were incorrectly predicted as malicious. The accuracy result for the CIC-IDS-2018 dataset is 99.73%, and the confusion matrix in Figure 4.41 indicates that the model correctly predicted 2076 malicious and 4432 benign instances. However, only one malicious instance was incorrectly identified as benign, while 16 benign instances were incorrectly identified as malicious. In addition, the CIC-IDS-2017 dataset achieved 99.84% accuracy, with the confusion matrix shown in Figure 4.42 showing that the model correctly predicted 6288

malicious and 7044 benign instances. On the other hand, the model incorrectly predicted seven malicious samples as benign and 13 benign samples as malicious. Table 4.9 details the criteria used to evaluate this model.

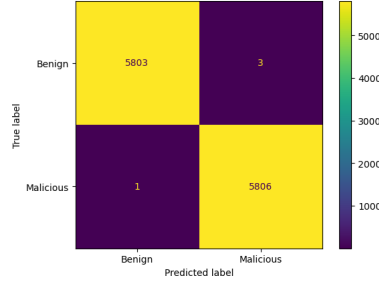


Figure 4.40: Confusion matrix on the CIC-MalMem-2022 dataset for XGBoost

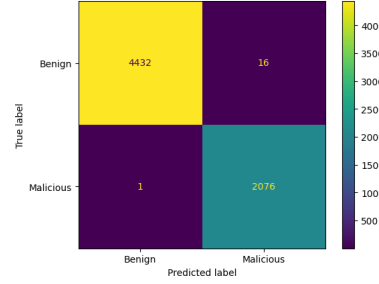


Figure 4.41: Confusion matrix on the CIC-IDS-2018 dataset for XGBoost

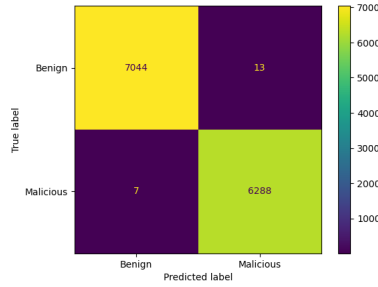


Figure 4.42: Confusion matrix on the CIC-IDS-2017 dataset for XGBoost

Table 4.9: Summary of XGBoost model results

Dataset	Accuracy	Precision	Recall	f1-score	CPU Us- age	Training Time	Testing Time
CIC-MalMem-2022	99.96%	99.98%	99.94%	99.96%	0.0000050	2.53 s	0.01 s
CIC-IDS-2018	99.73%	99.95%	99.23%	99.59%	0.0000057	14.46 s	0.03 s
CIC-IDS-2017	99.84%	99.88%	99.79%	99.84%	0.0000066	13.45 s	0.04 s

4.2.6.1 Learning Curve

Figure 4.43 shows that the training score achieved 100% at every cycle, while the cross-validation score was between 99.83 and 99.89% accurate. As shown in Figure 4.44, the CIC-IDS-2018 learning curve shows a high training score and high cross-validation score, and the training score does not plateau, indicating that the model is not overfitting the data during the training phase. According to Figure 4.45, the training score for the CIC-IDS-2017 dataset oscillates between 100% and 99.99%, in contrast to the cross-validation score that begins at 99.54% and increases to reach 99.90%.

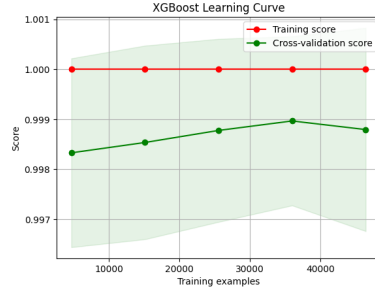


Figure 4.43: Learning curve on the CIC-MalMem-2022 dataset for XGBoost

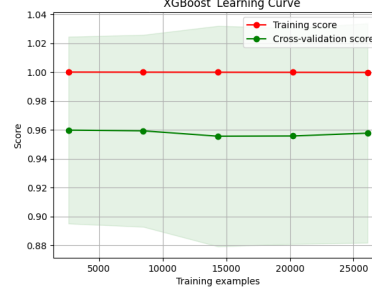


Figure 4.44: Learning curve on the CIC-IDS-2018 dataset for XGBoost

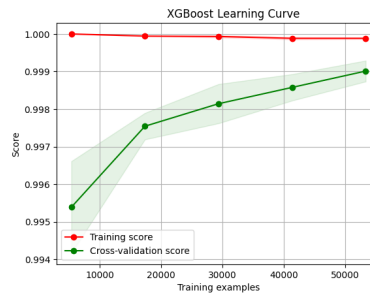


Figure 4.45: Learning curve on the CIC-IDS-2017 dataset for XGBoost

4.2.6.2 Hyperparametr in the Model

For the XGBoost model, the learning rate is a very important parameter. As the value of the parameter increases, overfitting may occur as a result. In this model, five different values have been assigned to the learning rate: 0.5, 0.7, 1, 2, and 10. In this graph, the x-axis represents the value of the learning rate, and the y-axis represents the mean test score. Figure 4.46, the best parameter for the CIC-MalMem-2022 dataset was 0.7 with a mean test score of 99.92%. According to Figure 4.47, the best parameter value for the CIC-IDS-2018 dataset is 0.5, with a mean test score of 99.50%. In addition, the best parameter value for the CIC-IDS-2017 dataset is 0.7, with a mean test score of 99.49% displayed in Figure 4.48.

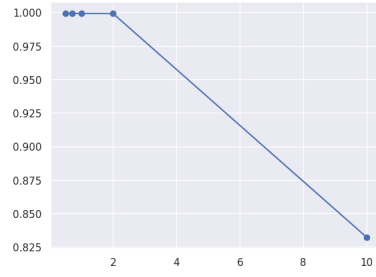


Figure 4.46: Hyperparameter tuning on the CIC-MalMem-2022 dataset for XGBoost

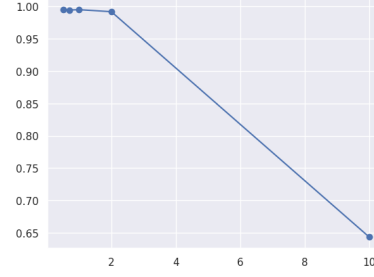


Figure 4.47: Hyperparameter tuning on the CIC-IDS-2018 dataset for XGBoost

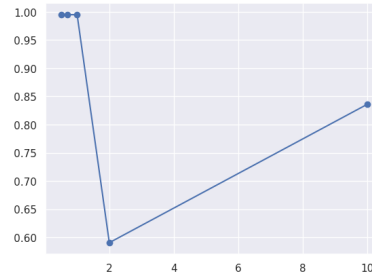


Figure 4.48: Hyperparameter tuning on the CIC-IDS-2017 dataset for XGBoost

4.3 Discussion

According to Table 4.10, Table 4.11, Table 4.12, and the requirement of the proposed architecture, the fastest model to detect malicious is XGBoost, while Stacking gives the highest accuracy in detecting malicious. Also Stacking model has less FN for all datasets.

Table 4.10: An analysis of the CIC-MalMem-2022 dataset for each model

Algorithm	Accuracy	Precision	Recall	f1-score	Testing Time
Adaboost	99.94%	99.96%	99.93%	99.94%	0.10 s
Voting	99.96%	99.98%	99.94%	99.96%	2.27 s
Stacking	99.98%	99.98%	99.98%	99.98%	2.90 s
SVM	99.93%	99.89%	99.98%	99.93%	0.44 s
XGBoost	99.96%	99.98%	99.94%	99.96%	0.01 s
RF	99.98%	99.98%	99.98%	99.98%	0.11 s

Table 4.11: An analysis of the CIC-IDS-2018 dataset for each model

Algorithm	Accuracy	Precision	Recall	f1-score	Testing Time
Adaboost	99.11%	99.37%	97.86%	98.61%	0.08 s
Voting	99.58%	100%	98.71%	99.35%	2.20 s
Stacking	99.61%	99.66%	99.13%	99.40%	1.64 s
SVM	97.41%	97.96%	93.91%	95.89%	1.56 s
XGBoost	99.73%	99.95%	99.23%	99.59%	0.03 s
RF	99.64%	99.90%	98.99%	99.44%	0.10 s

Table 4.12: An analysis of the CIC-IDS-2017 dataset for each model

Algorithm	Accuracy	Precision	Recall	f1-score	Testing Time
Adaboost	98.64%	98.18%	98.95%	98.56%	0.11 s
Voting	99.31%	98.91%	99.65%	99.28%	4.66 s
Stacking	99.86%	99.66%	99.74%	99.85%	2.93 s
SVM	94.04%	89.39%	99.14%	94.01%	9.81 s
XGBoost	99.84%	99.88%	99.79%	99.84%	0.04 s
RF	99.40%	98.96%	98.96%	99.36%	0.18 s

Due to the importance of testing time to this study, the experiment conducted by (Talukder et al., 2023), and (Patil et al., 2022). was re-conducted to determine the testing time. A comparison of this study's results and other studies is shown in Table 4.13, Table 4.14, and Table 4.15 for each dataset

used in this study. This study achieved 99.96% accuracy for the CIC-MalMem-2022 dataset within 0.01 seconds, while (Talukder et al., 2023) obtained 100% accuracy with a time of 0.08 seconds, which is higher than this study. Based on the CIC-IDS-2018 and CIC-IDS-2017 datasets, this study achieves greater accuracy and lower time than (Seth et al., 2021), (Patil et al., 2022). According to other studies, there is no global centralized traffic inspection before it is distributed to sub-organizations as a multi-model IDS based on the traffic load. The proposed architecture involves two layers of security. Using the multi-model IDS in the first layer inspects the traffic coming to the organization quickly. In contrast, IDSs in sub-organizations act as second-layer security, which is more accurate at detecting attacks.

Table 4.13: Comparative analysis of the CIC-MalMem-2022 dataset

Author	Feature Selection	Algorithm	Selected Feature	Accuracy	Testing Time
(Talukder et al., 2023)	XGBoost	RF	20	100%	0.09s
This Study	selectkbest	XGBoost	25	99.96%	0.01 s

Table 4.14: Comparative analysis of the CIC-IDS-2018 dataset

Author	Feature Selection	Algorithm	Selected Feature	Accuracy	Testing Time
(Seth et al., 2021)	RF&PCA	LightGBM	37	97.73%	17.94 s
This Study	Variance-Threshold	XGBoost	72	99.73%	0.03 s

Table 4.15: Comparative analysis of the CIC-IDS-2017 dataset

Author	Feature Selection	Algorithm	Selected Feature	Accuracy	Testing Time
(Patil et al., 2022)	Correlation heatmap techniques	Voting classifier	10	96.25%	15.86 s
This Study	selectkbest	XGBoost	30	99.84%	0.04 s

4.4 Limitations of Proposed Model

The multimodel based IDS was tested on a dataset of simulated traffic. Although the results are promising, we would need to collect a much larger dataset of real-world traffic and use techniques such as data augmentation to avoid overfitting. In addition, uses for the dataset for CIC-IDS-2017 uses only 400,000 records at the initial of the process, while the dataset for CIC-IDS-2018 uses only 50,000 records of the total records at the being. In contrast, all samples used for CIC-MalMem-2022 are used in this thesis. This study has conceptualized and presented the proposed architecture for multi-model, 'Multi-model IDS based on traffic load.' However, it is essential to note that the proposed architecture still needs to be implemented. The thesis aims to explore the potential advantages and feasibility of the proposed architecture without immediate practical implementation, with the primary focus being theoretical or conceptual. Also, conduct the thesis stepwise, beginning with a thorough understanding of theoretical aspects and feasibility before moving on to practical implementation in subsequent phases. Further, the primary objective is to evaluate the effectiveness of machine learning techniques for intrusion detection rather than implementing a specific architecture. A separate ML-based IDS was implemented and evaluated for the effectiveness of ML techniques. As part of this work, the aim was to evaluate the performance of the implemented ML-based IDS, although the

proposed architecture remains a theoretical construct. The proposed architecture may be implemented and evaluated in the future to verify its potential benefits compared to existing approaches.

4.5 Summary

This chapter presents a comprehensive analysis of the experimental outcomes derived from several machine learning models, including Adaboost, Voting classifier, Stacking classifier, SVM, RF, and XGBoost. The chapter also examines the proposed multi-model intrusion detection system (IDS) that utilizes traffic load as a basis for detection. This IDS uses machine learning techniques to identify and classify assaults. The suggested approach has a dual-layer security system, whereby incoming network traffic from the Wide Area Network (WAN) is first subjected to filtration by a firewall that operates on pre-established rules. Prior to reaching the central server, the traffic undergoes scrutiny by a multi-model Intrusion Detection System (IDS), serving as the first layer of protection. The selection of the IDS model at this layer is contingent upon the traffic load. In the event that the volume of traffic is above the predetermined threshold established by the network administrator, the most expedient model for identifying the assault will be activated. In the present research, it has been seen that the XGBoost model exhibits the greatest speed among the models considered. Furthermore, it has been determined that the model with superior accuracy is activated when the traffic falls below a certain threshold. Notably, the stacking classifier demonstrates the highest level of accuracy in identifying assaults. The second layer of security is implemented when the traffic is dispersed from the central server to the sub-organization. At this stage, the Stacking classifier is used to examine the traffic before its entry into the sub-organization. The suggested model exhibits novelty due to its imple-

mentation of global traffic inspection prior to ingress into the organization, as well as its use of a two-layer security approach. Additionally, the proposed model inspects the whole traffic with the faster model and inspects the subdomain always with high accuracy. XGBoost model in this thesis not only has high accuracy but is faster than other proposed IDS. Finally, the proposed IDS model considers time a critical parameter, while other related work does not consider it an important parameter.

Chapter 5

Conclusion and Recommendations

5.1 Overview

An overview of the research idea, results of the experiments, and limitations will be presented in this chapter.

5.2 Conclusion

The literature review examined various contemporary intrusion detection systems that utilize machine learning techniques across various environments. These techniques include information-based, similarity-based, probability-based, and error-based learning. The review also explored hybrid and software-defined networking (SDN) models in this context. The present study has undertaken comprehensive literature research, leading to development a novel intrusion detection system. This study presents an architecture design that ensures a global view of and inspection of the traffic coming to the various

organizational servers. The proposed IDS is designed to scan all traffic from the organization's servers and redirect it to the requested server by creating a centralized proxy server. So, the main objective achieved is the design of an architecture ensuring the global view and inspection of the traffic coming to the different organization servers as proposed in this thesis, and it is called Multi-model IDS based on traffic load. Furthermore, the second main objective of the study proposes creating machine-learning anomaly-based IDS with a high accuracy and detection rate and Intrusion Prevention Systems (IPS) prevention of malicious IPs through firewalls. This system incorporates many models and is primarily based on analyzing traffic load. Furthermore, it incorporates a two-layer security mechanism. Before reaching the central server, a preliminary layer of security will be implemented, including examining global traffic. The suggested Intrusion Detection System (IDS) would conduct the examination using two best models. So, the study presents and analyzes the results of Adaboost, Voting, Stacking, Support Vector Machine, Random Forest, and XGBoost models on the CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS- 2017 datasets. The results show that XGBoost and Stacking are the best models. The first model used is XGBoost, which exhibits rapid identification of harmful instances. Subsequently, a stacking classifier is utilized as the second model, demonstrating a notable level of accuracy but necessitating extended detection duration. The activation of each model will be determined based on the volume of traffic that enters the organization. The determination of the threshold value, which accurately represents the level of network traffic, is undertaken by the network administrator. The activation of the quickest model occurs when the amount of traffic is above the predetermined threshold value. Alternatively, the model with greater accuracy and longer detection times will be engaged. The second layer of security is implemented when the network traffic is dispersed from the central server to the sub-organization. In the subsequent

layer, the stacking model will be used to attain. They are achieving a high level of accuracy in detecting harmful activities. The findings of this experiment indicate that the XGBoost model exhibits the most speed in identifying malicious data across all three datasets (CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017). Conversely, the stacking approach demonstrates the greatest accuracy in this context. However, identifying fraudulent material necessitates a more significant amount of time. Based on the findings of the XGBoost model, the CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017 datasets exhibit accuracy rates of 99.96%, 99.73%, and 99.84%, respectively, achieved within time intervals of 0.01 seconds, 0.03 seconds, and 0.04 seconds respectively. The Stacking model achieved accuracy values of 99.98%, 99.61%, and 99.86% for the CIC-MalMem-2022, CIC-IDS-2018, and CIC-IDS-2017 datasets, respectively. The findings indicate that the multi-model intrusion detection system (IDS) that relies on traffic load is significantly capable of efficiently identifying and detecting malicious activity. The proposed model emphasizes the implementation of global traffic inspections before ingress into an organization, as well as the implementation of two-layer security strategies. The proposed model also inspects the entire traffic with a faster model and inspects the subdomains with high accuracy at all times. Compared to other IDS models proposed in this thesis, the proposed XGBoost model has high accuracy and efficiency. Although the proposed multimodel IDS is comparable to existing solutions in terms of accuracy, analyzing performance metrics reveals additional insights. The precision, F1 score, and recall of the system were carefully considered, showing its ability to identify true positives while minimizing false positives at the same time. In addition to its robust generalization across diverse datasets and simulator network traffic scenarios, the proposed multi-model IDS can adapt to varying threat landscapes, demonstrating its efficacy in dynamic environments. Also, the proposed IDS model considers time a

critical parameter, while other related work does not consider it an important parameter. Finally, this thesis introduces a multimodel intrusion detection system for a centralized network architecture that proves two-layer security and global inspection of the traffic entering the organization, further enriching the field of intrusion detection. It is believed that the discovery of these new aspects of cybersecurity signifies advancement in knowledge and the potential for future innovations.

5.3 Recommendations and Future Work

Per the suggestion, IPS prevention of malicious IPs through firewalls can be accomplished automatically using a threat intelligence feed. Also, the UNSW-NB15 dataset will be used. In our next research, we want to focus on the network architecture inside an authentic setting pertaining to a particular business. Subsequently, we will proceed to perform an empirical investigation within this real-world environment, wherein we will gather relevant traffic data. This data will then be appropriately processed and formatted to align with the models used in this study. Deep-learning algorithms will also be used in the subsequent phase.

References

- Abubakar, A., & Garko, A. B. (2021). A predictive model for network intrusion detection system using deep neural network. *Dutse J. Pure Appl. Sci.(DUJOPAS)*, 7(3a), 113–128.
- Ahmad, S., & Mir, A. H. (2021). Scalability, consistency, reliability and security in sdn controllers: a survey of diverse sdn controllers. *Journal of Network and Systems Management*, 29, 1–59.
- Ajila, S. A., Lung, C.-H., & Das, A. (2021). Analysis of error-based machine learning algorithms in network anomaly detection and categorization. *Annals of Telecommunications*, 1–12.
- Akshay Kumar, M., Samiayya, D., Vincent, P., Srinivasan, K., Chang, C.-Y., & Ganesh, H. (2022). A hybrid framework for intrusion detection in healthcare systems using deep learning. *Frontiers in Public Health*, 9, 824898.
- Alajanbi, M., Ismail, M. A., Hasan, R. A., & Sulaiman, J. (2021). Intrusion detection: A review. *Mesopotamian Journal of CyberSecurity*, 2021, 1–4.
- Aldallal, A., & Alisa, F. (2021). Effective intrusion detection system to secure data in cloud using machine learning. *Symmetry*, 13(12), 2306.
- Alharbi, A., & Alsubhi, K. (2021). Botnet detection approach using graph-based machine learning. *IEEE Access*, 9, 99166–99180.
- Aljanabi, M., Ismail, M. A., & Ali, A. H. (2021). Intrusion detection systems, issues, challenges, and needs. *International Journal of Compu-*

- tational Intelligence Systems*, 14(1), 560–571.
- Alzahrani, A. O., & Alenazi, M. J. (2021). Designing a network intrusion detection system based on machine learning for software defined networks. *Future Internet*, 13(5), 111.
- Amanoul, S. V., Abdulazeez, A. M., Zeebare, D. Q., & Ahmed, F. Y. (2021). Intrusion detection systems based on machine learning algorithms. In *2021 ieee international conference on automatic control & intelligent systems (i2caxis)* (pp. 282–287).
- Attou, H., Guezzaz, A., Benkirane, S., Azrour, M., & Farhaoui, Y. (2023). Cloud-based intrusion detection approach using machine learning techniques. *Big Data Mining and Analytics*, 6(3), 311–320.
- Azizan, A. H., Mostafa, S. A., Mustapha, A., Foozy, C. F. M., Wahab, M. H. A., Mohammed, M. A., & Khalaf, B. A. (2021). A machine learning approach for improving the performance of network intrusion detection systems. *Annals of Emerging Technologies in Computing (AETiC)*, 5(5), 201–208.
- Balyan, A. K., Ahuja, S., Lilhore, U. K., Sharma, S. K., Manoharan, P., Algarni, A. D., ... Raahemifar, K. (2022). A hybrid intrusion detection model using ega-pso and improved random forest method. *Sensors*, 22(16), 5986.
- Baniasadi, S., Rostami, O., Martín, D., & Kaveh, M. (2022). A novel deep supervised learning-based approach for intrusion detection in iot systems. *Sensors*, 22(12), 4459.
- Brownlee, J. (2020). *Data preparation for machine learning: data cleaning, feature selection, and data transforms in python*. Machine Learning Mastery.
- Chaganti, R., Suliman, W., Ravi, V., & Dua, A. (2023). Deep learning approach for sdn-enabled intrusion detection system in iot networks. *Information*, 14(1), 41.

- Chen, S., Wu, Z., & Christofides, P. D. (2021). Cyber-security of centralized, decentralized, and distributed control-detector architectures for nonlinear processes. *Chemical Engineering Research and Design*, 165, 25–39.
- Ding, H., Chen, L., Dong, L., Fu, Z., & Cui, X. (2022). Imbalanced data classification: A knn and generative adversarial networks-based hybrid approach for intrusion detection. *Future Generation Computer Systems*, 131, 240–254.
- Dini, P., & Saponara, S. (2021). Analysis, design, and comparison of machine-learning techniques for networking intrusion detection. *Designs*, 5(1), 9.
- Dutta, V., Choraś, M., Kozik, R., & Pawlicki, M. (2021). Hybrid model for improving the classification effectiveness of network intrusion detection. In *13th international conference on computational intelligence in security for information systems (cisis 2020) 12* (pp. 405–414).
- Elsayed, R. A., Hamada, R. A., Abdalla, M. I., & Elsaid, S. A. (2023). Securing iot and sdn systems using deep-learning based automatic intrusion detection. *Ain Shams Engineering Journal*, 102211.
- Géron, A. (2022). *Hands-on machine learning with scikit-learn, keras, and tensorflow*. ” O’Reilly Media, Inc.”.
- Hamida, S., El Gannour, O., Cherradi, B., Ouajji, H., & Raihani, A. (2020). Optimization of machine learning algorithms hyper-parameters for improving the prediction of patients infected with covid-19. In *2020 ieee 2nd international conference on electronics, control, optimization and computer science (icecocs)* (pp. 1–6).
- Hassan, I. H., Abdullahi, M., Aliyu, M. M., Yusuf, S. A., & Abdulrahim, A. (2022). An improved binary manta ray foraging optimization algorithm based feature selection and random forest classifier for network intrusion detection. *Intelligent Systems with Applications*, 16,

200114.

- Helmy, A., & Nayak, A. (2020). Centralized vs. decentralized bandwidth allocation for supporting green fog integration in next-generation optical access networks. *IEEE Communications Magazine*, 58(5), 33–39.
- Hnamte, V., Balram, G., et al. (2022). Implementation of naive bayes classifier for reducing ddos attacks in iot networks. *JOURNAL OF ALGEBRAIC STATISTICS*, 13(2), 2749–2757.
- Hnamte, V., & Hussain, J. (2023). Dcnnbilstm: An efficient hybrid deep learning-based intrusion detection system. *Telematics and Informatics Reports*, 100053.
- Izquierdo, C., Casas, G., Martin-Isla, C., Campello, V. M., Guala, A., Gkontra, P., ... Lekadir, K. (2021). Radiomics-based classification of left ventricular non-compaction, hypertrophic cardiomyopathy, and dilated cardiomyopathy in cardiovascular magnetic resonance. *Frontiers in Cardiovascular Medicine*, 1502.
- Jahwar, A. F., & Ameen, S. Y. (2021). A review on cybersecurity based on machine learning and deep learning algorithms. *Journal of Soft Computing and Data Mining*, 2(2), 14–25.
- Jaradat, A. S., Barhoush, M. M., & Easa, R. B. (2022). Network intrusion detection system: machine learning approach. *Indonesian Journal of Electrical Engineering and Computer Science*, 25(2), 1151–1158.
- Javeed, D., Gao, T., Khan, M. T., & Ahmad, I. (2021). A hybrid deep learning-driven sdn enabled mechanism for secure communication in internet of things (iot). *Sensors*, 21(14), 4884.
- Kelleher, J. D., Mac Namee, B., & D’arcy, A. (2020). *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT press.
- Khan, K., Mehmood, A., Khan, S., Khan, M. A., Iqbal, Z., & Mashwani,

- W. K. (2020). A survey on intrusion detection and prevention in wireless ad-hoc networks. *Journal of Systems Architecture*, 105, 101701.
- Kostinakis, K. (2022). A comparative evaluation of machine learning algorithms for the prediction of r/c buildings'seismic damage.
- Leevy, J. L., Hancock, J., Zuech, R., & Khoshgoftaar, T. M. (2020). Detecting cybersecurity attacks using different network features with lightgbm and xgboost learners. In *2020 ieee second international conference on cognitive machine intelligence (cogmi)* (pp. 190–197).
- Linardatos, P., Papastefanopoulos, V., & Kotsiantis, S. (2020). Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1), 18.
- Liu, C., Gu, Z., & Wang, J. (2021). A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning. *Ieee Access*, 9, 75729–75740.
- Louk, M. H. L., & Tama, B. A. (2022). Tree-based classifier ensembles for pe malware analysis: A performance revisit. *Algorithms*, 15(9), 332.
- Mirlekar, S., & Kanojia, K. P. (2022). A study on taxonomy and state-of-the-art intrusion detection system. *Available at SSRN 4157336*.
- Modi, T. M., & Swain, P. (2023). Hybrid deep learning models and link probability based routing in software defined-dcn. *The Journal of Supercomputing*, 1–24.
- Mohr, F., & van Rijn, J. N. (2022). Learning curves for decision making in supervised machine learning—a survey. *arXiv preprint arXiv:2201.12150*.
- Mussmann, S., Fang, A., Schmidt, L., & Jamieson, K. (2022). Data subset selection via machine teaching.
- Nagaraja, A., Boregowda, U., Khatatneh, K., Vangipuram, R., Nuvvusetty, R., & Kiran, V. S. (2020). Similarity based feature transformation for network anomaly detection. *IEEE Access*, 8, 39184–39196.

- Naveed, M., Arif, F., Usman, S. M., Anwar, A., Hadjouni, M., Elmannai, H., ... Umar, F. (2022). A deep learning-based framework for feature extraction and classification of intrusion detection in networks. *Wireless Communications and Mobile Computing*, 2022.
- Onah, J. O., Abdullahi, M., Hassan, I. H., Al-Ghusham, A., et al. (2021). Genetic algorithm based feature selection and naïve bayes for anomaly detection in fog computing environment. *Machine Learning with Applications*, 6, 100156.
- Ozkan-Okay, M., Samet, R., Aslan, Ö., & Gupta, D. (2021). A comprehensive systematic literature review on intrusion detection systems. *IEEE Access*, 9, 157727–157760.
- Patil, S., Varadarajan, V., Mazhar, S. M., Sahibzada, A., Ahmed, N., Sinha, O., ... Kotecha, K. (2022). Explainable artificial intelligence for intrusion detection system. *Electronics*, 11(19), 3079.
- Perez-Diaz, J. A., Valdovinos, I. A., Choo, K.-K. R., & Zhu, D. (2020). A flexible sdn-based architecture for identifying and mitigating low-rate ddos attacks using machine learning. *IEEE Access*, 8, 155859–155872.
- Ponmalar, A., & Dhanakoti, V. (2022). An intrusion detection approach using ensemble support vector machine based chaos game optimization algorithm in big data platform. *Applied Soft Computing*, 116, 108295.
- Priyadarsini, M., & Bera, P. (2021). Software defined networking architecture, traffic management, security, and placement: A survey. *Computer Networks*, 192, 108047.
- Pudjihartono, N., Fadason, T., Kempa-Liehr, A. W., & O’Sullivan, J. M. (2022). A review of feature selection methods for machine learning-based disease risk prediction. *Frontiers in Bioinformatics*, 2, 927312.
- Qazi, E. U. H., Faheem, M. H., & Zia, T. (2023). Hdlnids: Hybrid deep-

- learning-based network intrusion detection system. *Applied Sciences*, 13(8), 4921.
- Ramanan, A. (2022). Shallow learning vs deep learning.
- Ramirez-Hernandez, D., Stolwyk, R. J., Ownsworth, T., & Wong, D. (2021). A comparison of systematic instruction, error-based learning and trial and error to train the use of smartphone memory apps after acquired brain injury: A three-armed phase ii randomised controlled trial study protocol. *Brain Impairment*, 22(2), 217–232.
- Rincy N, T., & Gupta, R. (2021). Design and development of an efficient network intrusion detection system using machine learning techniques. *Wireless Communications and Mobile Computing*, 2021, 1–35.
- Saheed, Y. K., Abiodun, A. I., Misra, S., Holone, M. K., & Colomo-Palacios, R. (2022). A machine learning-based intrusion detection for detecting internet of things network attacks. *Alexandria Engineering Journal*, 61(12), 9395–9409.
- Samantaray, M., Satapathy, S., & Lenka, A. (2022). A systematic study on network attacks and intrusion detection system. In *Machine intelligence and data science applications: Proceedings of midas 2021* (pp. 195–210). Springer.
- Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3), 160.
- Sawant, S. S., & Prabukumar, M. (2020). A review on graph-based semi-supervised learning methods for hyperspectral image classification. *The Egyptian Journal of Remote Sensing and Space Science*, 23(2), 243–248.
- Seth, S., Singh, G., & Kaur Chahal, K. (2021). A novel time efficient learning-based approach for smart intrusion detection system. *Journal of Big Data*, 8(1), 1–28.

- Shetabi, M., & Akbari, A. (2020). Sahar: a control plane architecture for high available software-defined networks. *International Journal of Communication Networks and Distributed Systems*, 24(4), 409–440.
- Shrivastava, R., & Sisodia, D. S. (2019). Product recommendations using textual similarity based learning models. In *2019 international conference on computer communication and informatics (iccci)* (pp. 1–7).
- Soomro, A. M., Naeem, A. B., Ghafoor, M. I., Senapati, B., & Rajwana, M. A. (2023). A systematic review of artificial intelligence techniques used for ids analysis. *Journal of Computing & Biomedical Informatics*, 5(01), 52–67.
- Talukder, M. A., Hasan, K. F., Islam, M. M., Uddin, M. A., Akhter, A., Yousuf, M. A., ... Moni, M. A. (2023). A dependable hybrid machine learning model for network intrusion detection. *Journal of Information Security and Applications*, 72, 103405.
- Ubale, G., & Gaikwad, S. (2022). Sms spam detection using tfidf and voting classifier. In *2022 international mobile and embedded technology conference (mecon)* (pp. 363–366).
- Ullah, I., & Mahmoud, Q. H. (2021). Design and development of a deep learning-based model for anomaly detection in iot networks. *IEEE Access*, 9, 103906–103926.
- Unnisa A, N., Yerva, M., & MZ, K. (2022). Review on intrusion detection system (ids) for network security using machine learning algorithms. *International Research Journal on Advanced Science Hub*, 4(3), 67–74.
- Urrea, C., & Benítez, D. (2021). Software-defined networking solutions, architecture and controllers for the industrial internet of things: A review. *Sensors*, 21(19), 6585.
- Valdovinos, I. A., Pérez-Díaz, J. A., Choo, K.-K. R., & Botero, J. F. (2021).

- Emerging ddos attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions. *Journal of Network and Computer Applications*, 187, 103093.
- Verma, A., & Ranga, V. (2020). Machine learning based intrusion detection systems for iot applications. *Wireless Personal Communications*, 111, 2287–2310.
- Wang, X., Zhu, T., Xia, M., Liu, Y., Wang, Y., Wang, X., ... others (2021). Predicting the prognosis of patients in the coronary care unit via machine learning using xgboost. *Available at SSRN 3801858*.
- Xuan, D., Hu, H., Wang, B., & Liu, B. (2021). Intrusion detection system based on rf-svm model optimized with feature selection. In *2021 international conference on communications, computing, cybersecurity, and informatics (ccci)* (pp. 1–5).
- Yan, F., Wen, S., Nepal, S., Paris, C., & Xiang, Y. (2022). Explainable machine learning in cybersecurity: A survey. *International Journal of Intelligent Systems*, 37(12), 12305–12334.
- Yu, T., & Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*.
- Zhang, H., Lan, W., & Zhang, D. (2022). Anomaly intrusion detection of wireless communication network-based on markov chain model. *Security and Communication Networks*, 2022.
- Zhang, Y., Wang, H., Chen, W., Zeng, J., Zhang, L., Wang, H., & Weinan, E. (2020). Dp-gen: A concurrent learning platform for the generation of reliable deep learning based potential energy models. *Computer Physics Communications*, 253, 107206.
- Zhao, R., Mu, Y., Zou, L., & Wen, X. (2022). A hybrid intrusion detection system based on feature selection and weighted stacking classifier. *IEEE Access*, 10, 71414–71426.
- Zhao, Y., & Wang, Z. (2022). Subset simulation with adaptable intermedi-

ate failure probability for robust reliability analysis: An unsupervised learning-based approach. *Structural and Multidisciplinary Optimization*, 65(6), 172.

ملخص الدراسة

أنظمة كشف التسلل (IDS) هي جزء أساسي من الأمن السيبراني. وظيفتها الأساسية هي مراقبة حركة مرور الشبكة لتحديد أي نشاط ضار وتصنيفه. في الوقت الحاضر، تم التشكيك في فعالية أنظمة كشف التسلل التقليدية (IDS) بسبب انتشار الإنترنت بشكل كبير، مما أدى إلى ظهور أشكال جديدة من الهجمات السيبرانية. ظهرت تقنيات التعلم الآلي كبديل قابل للتطبيق لتعزيز أنظمة كشف التسلل التقليدية (IDS). ستفحص هذه الدراسة الاختلافات في حركة المرور الواردة لأنظمة كشف التسلل للشبكات (NIDS) بسبب تثبيتها في شبكات فرعية مختلفة داخل منظمة. في بعض الحالات، يُرى أن مهاجمًا واحدًا قد يوجه جهوده إلى خوادم مختلفة تابعة لمؤسسات مختلفة. غالبًا ما يُنظر إلى هذا السلوك من قبل أنظمة كشف التسلل (IDS) على أنه هجمات غير متكررة، مما يقلل من فعالية الكشف. الهدف هو تطوير نموذج لنظام كشف التسلل (IDS) مدعوم بالتعلم الآلي قادر على تحديد وتصنيف حركة مرور الشبكة الضارة المحتملة الواردة عبر واجهات مختلفة داخل الشبكات التنظيمية. يتضمن النهج المقترح إنشاء خادم وكيل مركزي سيعمل كمستقبل لجميع حركة المرور الواردة الموجهة إلى خوادم المنظمة. سيزود هذا الخادم الوكيل بنظام كشف التسلل متعدد النماذج (IDS) مصمم لتحليل حمولة حركة المرور. بعد التحليل، سيعيد توجيه الخادم الوكيل حركة المرور إلى الخادم المطلوب. للمشروع هدفان رئيسيان: بناء بنية تضمن منظورًا شاملاً، فحص حركة المرور الواردة الموجهة إلى خوادم مختلف المنظمات. وتطوير نظام كشف التسلل (IDS) قائم على التعلم الآلي واكتشاف الأنماط بمعدل دقة ومعدل اكتشاف مرتفعين. سيتم تقييم نظام كشف التسلل (IDS) القائم على التعلم الآلي (ML) قيد الدراسة جنبًا إلى جنب مع النماذج الأخرى المقترحة في الدراسات الأدبية القائمة، مع التركيز على تقييم نسب الدقة ومعدلات الاكتشاف الخاصة بهم. تضمنت تقييم نظام كشف التسلل (IDS) المقترح استخدام ثلاث مجموعات بيانات مختلفة، وهي مجموعات بيانات CIC-MalMem-2022 وCIC-IDS-2018 وCIC-IDS-2017 أظهر نموذج XGBoost أداء استثنائيًا في الكشف السريع، حيث حقق معدلات دقة تبلغ 99.96% و99.73% و99.84% لمجموعات بيانات CIC-MalMem-2022 وCIC-IDS-2018 وCIC-IDS-2017 على التوالي. تم الحصول على هذه النتائج المبهرة في فترات زمنية قصيرة جدًا تبلغ 0.01 ثانية و0.03 ثانية و0.04 ثانية على التوالي. لدى نموذج Stacking أعلى مستوى من الدقة بين النماذج المُقيّمة. على وجه التحديد، يحقق دقة تبلغ 99.98% لمجموعة بيانات CIC-MalMem-2022، و99.61% لمجموعة بيانات CIC-IDS-2018، و99.86% لمجموعة بيانات CIC-IDS-2017. أظهر نظام كشف التسلل (IDS) الذي تم تطويره نتائج أفضل من حيث الدقة ووقت الكشف مقارنة بالدراسات السابقة في هذا المجال.

جامعة البحرين

كلية تقنية المعلومات



تتصميم وتحليل بنية فعالة لأنظمة اكتشاف
الاختراقات في الشبكات المستندة إلى التعلم الآلي
أطروحة مقدمة كجزء من متطلبات الحصول على
درجة الماجستير في الأمن الإلكتروني

إعداد

نورة سعد الرميحي

٢٠٠٥٣٠١٩

إشراف

د. محسن الرواشد

(أستاذ مشارك)

جامعة البحرين

مملكة البحرين

ديسمبر ٢٠٢٣