Instituto Politécnico
de Viana do Castelo

AN ARCHITECTURE TO GENERATE CLASSIFIED DATASETS AND IMPROVE PERFORMANCE OF INTRUSION DETECTION SYSTEMS

Diogo Francisco Rodrigues Teixeira

Escola Superior de Tecnologia e Gestão

Instituto Politécnico
de Viana do Castelo

Diogo Francisco Rodrigues Teixeira

AN ARCHITECTURE TO GENERATE CLASSIFIED DATASETS AND IMPROVE PERFORMANCE OF INTRUSION DETECTION SYSTEMS

Nome do curso de Mestrado

Mestrado em Cibersegurança

Trabalho efetuado sob a supervisão de

Professor Pedro Filipe Cruz Pinto

Professor Silvestre Lomba Malta

Abril de 2022

Mestrado em
Cibersegurança
Master in
Cybersecurity

An Architecture to Generate Classified Datasets and Improve Performance of Intrusion Detection Systems

a master's thesis authored by

Diogo Francisco Rodrigues Teixeira

and supervised by

Pedro Filipe Cruz Pinto

Professor Adjunto, IPVC

Silvestre Lomba Malta

Professor Assistente, IPVC

This thesis was submitted in partial fulfilment of the requirements for the Master's degree in Cybersecurity at the Instituto Politécnico de Viana do Castelo

ipvc

19 of April, 2022

Abstract

Nowadays, a set of services are available online with various associated data. It is essential to ensure the availability, integrity and confidentiality of all data. However, cyberattacks are a major threat. In this sense, an Intrusion Detection System (IDS) is an important tool to prevent potential threats to systems and data.

It is necessary to implement new mechanisms with intelligence to successfully defend the complexity and intelligence of attacks, that is, to increase their efficiency.

Anomaly-based IDSs may deploy machine learning algorithms to classify events either as normal or anomalous and trigger the adequate response. When using supervised learning, these algorithms require classified, rich, and recent datasets. Thus, to foster the performance of these machine learning models, datasets can be generated from different sources in a collaborative approach, and trained with multiple algorithms.

This document proposes a vote-based architecture to generate classified datasets and improve performance of supervised learning-based IDSs. In a regular basis, multiple IDSs in different locations (companies) send their logs to a central system that combines and classifies them using different machine learning models and a majority vote system. Then, it generates a new and classified dataset, which is trained to obtain the best updated model to be integrated into the IDS of the companies involved. In this way, intrusion detection systems are frequently updated with the best machine learning model to increase their efficiency.

The proposed architecture trains multiple times with several algorithms and, to shorten the overall runtimes, the proposed architecture was deployed in Fed4FIRE+, a federated testbed, with Ray to distribute the tasks by the available resources. This implementation allowed a reduction of the time in the classification between 31% and 33%, and in the training time of 43%.

A set of machine learning algorithms and the proposed architecture were assessed. When compared with a baseline scenario, the proposed architecture enabled to increase the accuracy by 11.5% and the precision by 11.2%.

Keywords: Machine Learning. Supervised Learning. Classified Datasets. Voting System. Multitasking. Distributed. Training Time. IDS.

Resumo

Hoje em dia, um conjunto de serviços estão disponíveis em rede com vários dados associados. É essencial garantir a disponibilidade, integridade e confidencialidade de todos os dados. Contudo, os ataques informáticos são uma grande ameaça. Neste sentido, um Sistema de Deteção de Intrusão (IDS) é uma ferramenta importante para prevenir potenciais ameaças a sistemas e dados.

É necessário implementar novos mecanismos com inteligência para defender com sucesso a complexidade e inteligência dos ataques, isto é, aumentando a sua eficiência.

Os sistemas de deteção de intrusão baseados em anomalias podem implementar algoritmos de *machine learning* para classificarem eventos como normais ou anómalos e acionar a resposta adequada. Ao utilizar aprendizagem supervisionada, os algoritmos requerem conjuntos de dados (*datasets*) classificados, enriquecidos e recentes. Assim, para fomentar o desempenho desses modelos de *machine learning*, conjuntos de dados podem ser criados em tempo real com registos de diferentes origens numa abordagem colaborativa e treinados por vários algoritmos.

Este trabalho propõe uma arquitetura baseada num sistema de votação para criar conjuntos de dados classificados e melhorar o desempenho dos sistemas de deteção de intrusão baseados em aprendizagem supervisionada. Em tempo real, vários sistemas de deteção de intrusão em diferentes locais (empresas) enviam os seus registos para um sistema central que os combina e classifica usando diferentes modelos de *machine learning* e um sistema de votação por maioria. Em seguida, cria um novo conjunto de dados classificados que é treinado para obter o melhor modelo atualizado que será integrado nos sistemas de deteção de intrusão das diferentes empresas envolvidas. Desta forma, os sistemas de deteção de intrusão são frequentemente atualizados com o melhor modelo de *machine learning* para aumentarem a sua eficiência.

A arquitetura proposta treina várias vezes com vários algoritmos e, para diminuir os tempos de execução, a arquitetura proposta foi implementada no Fed4FIRE+ com Ray a gerir a distribuição das tarefas pelos recursos disponíveis. Esta implementação permitiu uma redução do tempo na classificação entre 31% e 33%, e no tempo de traino de 43%.

Neste trabalho a arquitetura proposta foi avaliada com vários algoritmos de machine learning. Quando comparada com um cenário de base (um único sistema de deteção de intrusão), a arquitetura proposta aumentou a exatidão em 11.5% e a precisão em 11.2%.

Palavras-chave: Aprendizagem Supervisionada. Conjuntos de dados. Classificação. Sistema de Votos. Multitarefa. Sistema distribuído. Tempo de treino. Sistema de Deteção de Intrusão.

Acknowledgements

The conclusion of this thesis and the arrival at the desired level depended on the important contribution of countless people and institutions. So, I want to thanks:

To the supervisors Professor Pedro Pinto and Professor Silvestre Malta for their support, help and total dedication in monitoring the thesis, as well as in the advice provided in order to complement the requirement involved in it.

To the Instituto Politécnico de Viana do Castelo (IPVC), in particular the Escola Superior de Tecnologia e Gestão (ESTG) for the opportunity to enter and enrich my knowledge in the Masters in Cybersecurity.

To the entire group of professors of the masters who have always encouraged me along my work. Also, thanks to my master's colleagues and friends for their support.

To my girlfriend who has always accompanied me, understood and encouraged me throughout this important stage.

To my parents and my family for the values they transmitted to me as well as for the constant monitoring and encouragement for the success of this work.

And to all those who were directly and indirectly involved in this journey.

Contents

List of Figures **viii**

List of Tables **ix**

List of Listings **xi**

List of Abbreviations **xii**

1 Introduction **1**

1.1 Context 2

1.2 Problem Statement and Motivation 2

1.3 Objectives 3

1.4 Contributions 3

1.5 Organization 4

2 Background **5**

2.1 Intrusion Detection Systems 5

2.2 Machine Learning 7

3 Related Work **10**

3.1 Intrusion Detection Systems and Machine Learning 10

3.2 Scalable Testbed Platforms and Tasks Ditribution 12

4 The Proposed Architecture **13**

5 Validation **21**

5.1 Dataset Selection and Preparation 21

5.2 Machine Learning Algorithms Selection 25
5.3 Performance Results and Analysis 33

6 Conclusions **45**

References **47**

List of Figures

1.1 Envisioned Scenario 3

2.1 Types of IDSs 5

2.2 Confusion Matrix 9

4.1 Overview of the Proposed Architecture 14

4.2 Detailed operation of the Proposed Architecture 15

4.3 Initial operation of the Proposed Architecture 16

4.4 Unclassified logs collection from companies IDSs 17

4.5 Detail of the Master's internal procedures 18

4.6 Master's internal procedures for classification, voting and dataset generation 19

4.7 Training, selection and deployment phases 20

5.1 Proposed Architecture validation scenario 35

5.2 Results of accuracy and precision for the Baseline and the Proposed Architecture scenarios 37

5.3 Implementation of the Proposed Architecture using Fed4FIRE+ and Ray . 38

5.4 Results of time to classify subsets in one server and in multiple servers . . . 41

5.5 Results of time to training subsets in one server and in multiple servers . . 44

List of Tables

5.1 Description of each label and number of instances 22

5.2 Description of each select feature 24

5.3 Results of the accuracy of each algorithm for each subset 26

5.4 Results of the precision of each subset for each tested algorithm 27

5.5 Results of the accuracy of each algorithm for each subset after applying oversampling 30

5.6 Results of the precision of each subset after applying oversampling for each tested algorithm 31

5.7 Elapsed runtime results for each algorithm and per subset 32

5.8 Elapsed runtime results for each algorithm and per subset after applying oversampling 33

5.9 Accuracy and Precision for Sample #0 34

5.10 Models used for the Baseline and the Proposed Architecture scenarios . . . 36

5.11 Accuracy results of Baseline and Proposed Architecture scenarios 36

5.12 Precision results of Baseline and Proposed Architecture scenarios 36

5.13 Time to classify records on single server and on multiple servers (Sample #1) 39

5.14 Time to classify records on single server and on multiple servers (Sample #2) 39

5.15 Time to classify records on single server and on multiple servers (Sample #3) 39

5.16 Time to classify records on single server and on multiple servers (Sample #4) 40

5.17 Variation of the time used for classification 40

5.18 Time to train subsets on single server and on multiple servers (Sample #1) 42

5.19 Time to train subsets on single server and on multiple servers (Sample #2) 42

5.20 Time to train subsets on single server and on multiple servers (Sample #3) 42

5.21 Time to train subsets on single server and on multiple servers (Sample #4) 43

5.22 Variation of the time used for training 43

List of Listings

5.1 Script for the Selection of relevant features 23
5.2 Applying SMOTE to balance the dataset 28

List of Abbreviations

ACL	Access Control Lists
AI	Artificial Intelligence
AUC	Area Under Curve
DDoS	Distributed Denial of Service
DTC	Decision Tree Classifier
FN	False Negative
FP	False Positive
GENI	Global Environment for Network Innovations
GPU	Graphic Processing Unit
HIDS	Host-Based Intrusion Detection System
IDS	Intrusion Detection System
IPS	Intrusion Prevention Systems
KNN	K-Nearest Neighbors
MCyber	Master in Cybersecurity
ML	Machine Learning
NIDS	Network-based Intrusion Detection System

OSSEC	Open Source Host-based Intrusion Detection System
RFC	Random Forest Classifier
RL	Reinforcement Learning
SAVI	Smart Applications on Virtual Infrastructure
SIEM	Security Information and Event Management
SL	Supervised Learning
SLog	Simple Logistics
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UL	Unsupervised Learning

Chapter 1

Introduction

The digital transformation provides businesses with a wide range of advantages and resources, and according to [1], data processed and exchanged in computer and networks is the most valuable resource. Thus, the digital transformation has also opened new avenues for criminal activities. Cyberattacks are persistently performed against companies and institutions and these criminal activities may have different objectives such as to disrupt services, steal confidential information, or perform extortion [2]. According to a Accenture report [3], in 2021 there was an increase in the average of attacks per company of 31% compared to 2020. On the other hand, there is an 82% increase in investment in cybersecurity compared to 2020.

The complexity and intelligence of attacks evolves and, therefore, defense systems need to keep the pace to be effective [4]. To prevent or mitigate the impact of these attacks, several tools and systems can be implemented such as firewalls, Security Information and Event Management (SIEM), Access Control Lists (ACL), Intrusion Detection System (IDS), Intrusion Prevention Systems (IPS), among others.

This thesis presents the details of a proposed architecture to improve the performance of IDS in a collaborative environment. The following sections present the context, the motivation, the objective, and the contributions resulting from this work.

1.1 Context

An IDS is an important tool for a system administrator to prevent potential threats to systems and data, as it aims to detect attacks against information systems and protect these systems against malware and unauthorized access to a network or a system [5]. IDSs monitor a network or system and their detection method can be classified as signature-based or anomaly-based IDS.

A signature-based IDS compares the monitored events against a pre-programmed list of known threats/signatures and their indicators of compromise. An anomaly-based IDS classifies the events either as normal or anomalous, according to an expected behavior or pattern. In this method, the detection is triggered when the networks' or systems' behavior does not follow the normal behavior or pattern previously defined [6].

The patterns of anomaly-based IDSs can be tested using Machine Learning (ML) algorithms. ML is a subset of the Artificial Intelligence (AI) area that uses data and data analysis based on iterative algorithms to learn from this data. The learning process of these iterative ML algorithms may be supervised, i.e. in Supervised Learning, the algorithm learns from experience by using labeled datasets to be trained to classify or predict outcomes accurately.

1.2 Problem Statement and Motivation

The IDSs are an important tool against cyberattacks, and they must be constantly improving their performance and protection levels. A given company wishing to protect its networks, devices or services with an anomaly-based IDS may use a supervised learning algorithm. However, for this algorithm to be effective, it must count on a previously classified and updated dataset, which is not simple to obtain or create. Classified datasets already available are not updated regularly and the use of automated tools to classify a dataset introduces errors and requires, to some extent, human intervention. On the other hand, a dataset that only contains the company's own records will not allow the model or models to learn from different environments, i.e. other companies. Thus, the effectiveness of these algorithms can be improved if the supervised model uses richer datasets containing records of different realities in a collaborative approach, and trained with multiple algorithms.

1.3 Objectives

The main objective of this project is to improve the performance of anomaly-based IDSs in a collaborative environment. Fig 1.1 presents the envisioned scenario. It assumes that a centralized architecture collects logs from the IDSs of multiple companies, generates classified datasets using a vote-based approach, trains and selects the best machine learning algorithms, and finally deploys the most accurate and updated machine learning model to each companies' IDS. Thus, the IDSs of each company will have the best machine learning models trained with updated records from multiple environments, to increase their performance to detect attacks on the premises of each company.

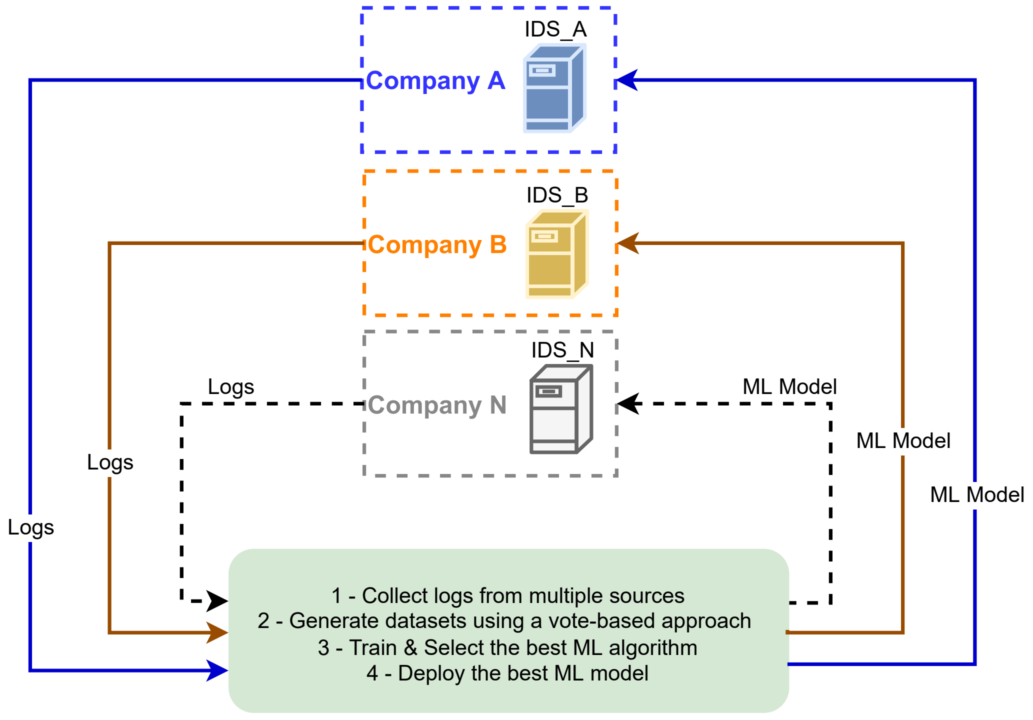

IDS_A
Company A
IDS_B
Company B
IDS_N
Company N
Logs
Logs
Logs
ML Model
ML Model
ML Model
1 - Collect logs from multiple sources
2 - Generate datasets using a vote-based approach
3 - Train & Select the best ML algorithm
4 - Deploy the best ML model

Figure 1.1: Envisioned Scenario

1.4 Contributions

This thesis provides a vote-based architecture to generate classified datasets and improve performance of Supervised Learning-based Intrusion Detection Systems. With the proposed architecture, classified datasets are created in real time by a voting system with records, i.e. services logs, from different environments (IDSs located in different companies). The classified datasets are trained by different models to generate new models. All models are deployed to the voting system and the best model is sent to the IDSs of companies. The aggregation of recent records from different realities enriches the dataset, increasing the accuracy and precision of the IDS over time, thus increasing the efficiency in protecting and preventing attacks.

This work resulted in the following publications:

1. Diogo Teixeira; Silvestre Malta; Pedro Pinto. **An Architecture using Ray to Distribute Tasks in a Federated Testbed Platform to Reduce Machine Learning Training Time**. Abstract - SASYR – 1st Symposium of Applied Science for Young Researchers, 7 July 2021, online, Portugal. URL: `http://hdl.handle.net/10198/23849`

2. Diogo Teixeira; Silvestre Malta; Pedro Pinto. **A Vote-Based Architecture to Generate Classified Datasets and Improve Performance of Intrusion Detection Systems Based on Supervised Learning**. Future Internet Journal, 2022, 14, 72. `https://doi.org/10.3390/fi14030072`

1.5 Organization

This document is organized as follows. In Chapter 2 the background concerning this work is presented. In Chapter 3 the related work is presented. Chapter 4 details the proposed architecture and its operation. In Chapter 5 presents the dataset selection and preparation, the machine learning algorithms selection, the validation tests, and the results and analysis of the proposed architecture. In Chapter 6 the conclusions are presented.

Chapter 2

Background

This chapter presents an overview of the relevant foundations for the current work, namely regarding Intrusion Detection Systems and Machine Learning.

2.1 Intrusion Detection Systems

An IDS protects corporations and institutions against cyberattacks by monitoring multiple events in different agents and triggering alerts or actions [7]. Fig 2.1 presents the different types of IDSs according to main categories.

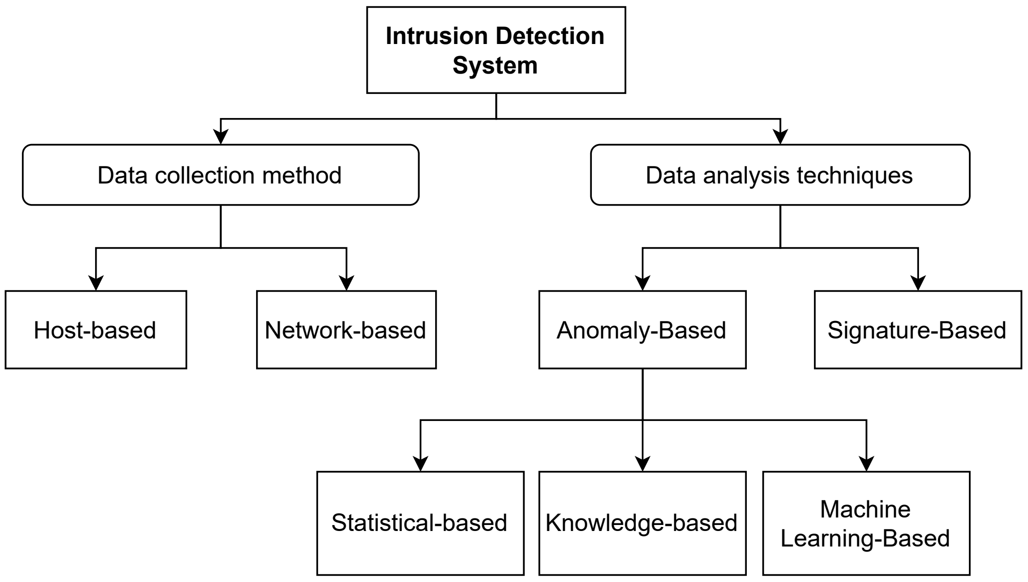


Figure 2.1: Types of IDSs

Regarding data collection method, IDSs can be classified into two different types: the

Host-Based Intrusion Detection System (HIDS) and Network-based Intrusion Detection System (NIDS). The HIDS has capabilities of monitoring and analyzing the internals of a computing system for malicious activity. The NIDS operates in a network where packets are captured and analyzed to enable the implementation of adequate protective measures. In [8] it is presented a detailed study of these two types. It is concluded that the best choice depends on purposes, risks and features.

Regarding the data analysis techniques, IDS can be categorized as signature-based or anomaly-based. The signature-based IDS is used for known patterns or threats. The network traffic is scanned to detect attack patterns, in form of signatures, and identify the intrusion. These IDSs are effective when detecting attack for which it has already a signature, however, attack variants can be designed to circumvent the detection process of these IDSs. The anomaly-based IDS uses mechanisms to characterize normal usage behaviors and recognizes departures from normal as potential intrusions. Thus, it is able to detect attempts to exploit new and unforeseen vulnerabilities. The Kumar et al. [9] concluded that the anomaly-based IDSs should be flexible, fast, dynamic to recognize new patterns and find the reason for false alarms. In [10], a performance evaluation of these modes is presented. The conclusion presented shows that both methods have limitation to detect known and unknown attacks. So it is necessary to implement new mechanisms to increase the effectiveness of IDS.

The anomaly-based IDS can be classified into three different types: Statistical-based, Knowledge-based, and Machine Learning-Based. The Statistical-based to build the normal pattern, statistical properties such as mean and variance are used. Thus, the normal patterns are calculated and a score is assigned to records that deviate from these patterns. If this score reaches the threshold (based on the number of events over a period of time), an alarm will be triggered. The Knowledge-based IDSs are used to extract knowledge of specific system vulnerabilities and attacks. Then intrusions or attacks can be identified with this knowledge. The machine learning-based IDS is the ability to learn and improve performance over time. Thus, based on previous results or recently acquired data, with training, the system can improve its execution.

2.2 Machine Learning

With AI, systems perceive the environment around them and are capable of learning, creativity, reasoning and planning to solve a given problem. The ML explores the learning process of AI. ML refers to the system that has the ability to learn and to build on that learning to predict results and make decisions. In this sense, with the execution of algorithms, these systems learn based on the data sets, or previous data, that are presented to them.

The learning process of machine learning algorithms can be classified as Unsupervised Learning (UL), Supervised Learning (SL), or Reinforcement Learning (RL). These processes can be detailed as follows:

- Supervised Learning:

  In this model, a set of classified data are needed. In this way, the algorithm knows the true classification of the input, and as a result, learns what decisions to make in the future. The main disadvantage in supervised learning focuses on its low efficiency when data changes patterns. Thus, it is usually applied to assess risk, forecast sales and detect SPAM.

- Unsupervised Learning:

  The model learns patterns from an unclassified dataset. It focuses on looking for similarities between inputs for future actions. The great advantage of Unsupervised Learning is due to the fact that it does not need classified data, that is, it does not need manual intervention. Thus, with the ability to discover patterns for grouping data, it is the ideal solution for exploratory data analysis, customer segmentation, anomaly detection and image recognition.

- Reinforcement Learning:

  In Reinforcement Learning, the agent interacts with the environment to make decisions and receive feedback. Thus, if the decision made by the agent is the right one receives a reward, while if he makes a mistake, he is penalized. In this way, the greatest reward is always sought, to achieve the goal. With its ability to learn and obtain the best possible reward, it is an algorithm widely used in games, self driving cars and Natural Language Processing (NLP).

There are several machine learning algorithms, and to evaluate the performance of each one, there are a set of evaluation factors that can be defined as follows:

- True Positive (TP): Number of predicts that model correctly predicts the positive class.
- True Negative (TN): Number of predicts that model correctly predicts the negative class.
- False Positive (FP): Number of predicts that model incorrectly predicts the positive class.
- False Negative (FN): Number of predicts that model incorrectly predicts the negative class.

These evaluation factors can be used by performance evaluation metrics or classification metrics that can be defined as follows:

- Precision: The proportion between true positives and the sum of true positives and false positives, as presented in Equation 2.1.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.1}$$

- Accuracy: The ration of the sum of true positives and true negatives to the total number of classifications, as presented in Equation 2.2.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.2}$$

- Recall: The proportion between true positives and the sum of true positives and false negatives, as presented in Equation 2.3.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.3}$$

- F1-score: The harmonic mean of precision and recall, as presented in Equation 2.4.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

- Confusion Matrix: Distinguishes true positive, false positive, true negative and false negative predictions, as presented in Fig. 2.2.

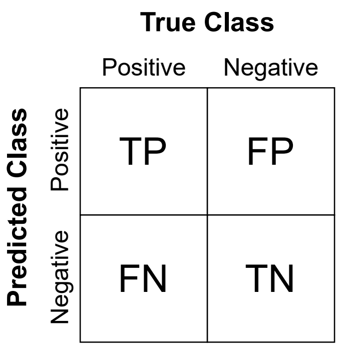


Figure 2.2: Confusion Matrix

Chapter 3

Related Work

In this chapter is presented a review on related works regarding intrusion detection systems and machine learning integration, and on related works regarding scalable testbed platforms and tasks distribution.

3.1 Intrusion Detection Systems and Machine Learning

As cyberattacks are constantly evolving, so IDS must also progress to be efficient [11]. A set of research works are focused on improving the operation and performance of the IDS. Authors in [12] propose an architecture to reduce the false alarm rate of the attack detection. Authors in [13] propose to prevent Distributed Denial of Service (DDoS) attacks by using the configuration features and rules adjustments of Open Source Host-based Intrusion Detection System (OSSEC) and describe the operation of an algorithm used to distinguish real of false DDoS alerts. In order to protect a set of machines, Teixeira et al. [14] implemented a platform to override false-positives and false-negatives of OSSEC IDS. The proposed platform is able to apply an override action in multiple agents, saving human intervention time.

Voting based systems are also a research line that is used to improve the detection performance of an IDS. Authors in [15] propose an ensemble adaptive voting algorithm. The results show that the final accuracy is improved with the use of an adaptive voting algorithm. Panda et al. [16] designed a voting system to detect errors and concluded that it performs efficiently in terms of high detection rate and low false positive rate. Authors in [17] and [18] have implemented a voting system that uses probability mechanisms to define the final classification. Mahfouz et al. [19] propose an ensemble classifier model that include a voting system to improve detection accuracy and true positive rate, and decrease the false positive rate. Authors in [20] propose a voting system to decrease false alarms. The results show that, when compared different deep learning models, false alarms can be reduced up to 75%.

Machine learning can be used to adapt an IDS to the dynamic and complex nature of the attacks [21]. According to Haripriya et al. [22], the main objective of applying machine learning algorithms in an IDS, focuses on obtaining a low false alarm rate and a high detection rate. As highlighted in [23], using machine learning techniques in an IDS can reduce the occurrence of false positives. Authors also point that one or more models should be used to increase the performance of their detection. In [24], Vikram et al. implement unsupervised learning algorithms in an IDS. The results demonstrate better efficiency compared to IDS without machine learning, with an Area Under Curve (AUC) score of 98.3%. In [25], Anthi et al. propose a IDS that uses a supervised approach to detect a range of popular network based cyberattacks. The performance of the proposal is greater than 90% and can successfully distinguish between malicious or benign activity. In [26], authors developed a supervised machine learning system on IDS to classify network traffic whether it is malicious or benign, with a detection rate of 94.02%. In [27], four algorithms are compared to implement on IDS and all of them have accuracy greater than 96%. In [28], Rani et al. propose an efficient method with uniform detection system based on supervised machine learning technique, that obtained an accuracy of 99.9%. In [29], authors implement a collaborative multi-agent reinforcement learning to make the detection more efficient. The results are better in comparison with baseline approach. In [30], Latif et al. proposed a Dense Random Neural Network (DnRaNN) technique to detect attacks in an IoT environment. The authors obtained an accuracy of 99,14% when using binary classification and 99,05% when using multiclass scenarios. In [31], Kunal et al. presented and compared various machine learning methodologies applied in IDS. Thus, it is demonstrated that the efficiency of the algorithms used will vary according to the final objective. To the best of our knowledge, none of the works presented previously proposed a vote-based architecture of multiple IDS to generate new datasets and improve their performance in a collaborative approach.

3.2 Scalable Testbed Platforms and Tasks Ditribution

The implementation of machine learning algorithms requires time and resources to train. As highlighted by Mo et al. [32], a long training time can add significant costs. The authors in [33] highlights that the process of building and deploying Machine Learning models includes several phases and the training phase is taken as one of the most time-consuming. Authors in [34] refer that long training running time of the models leads to the use of distributed systems for an increase of parallelization and total amount of I/O bandwidth. Thus, in this context it can be considered the use of federated testbed platforms. These platforms enable the implementation, validation, and testing architectures in a scalable environment. In particular testbeds such as Fed4FIRE+ [35], Global Environment for Network Innovations (GENI) [36] and Smart Applications on Virtual Infrastructure (SAVI) [37] can be considered, as these platforms have a large system capacity and a rich set of experimentation services.

To distribute tasks and enable parallel processing, multiple tools can be used. Authors in [38] surveyed and tested the following open source python-based libraries for parallel processing: Ray [39], Ipyparallel [40], Dispy [41], Pandaral-lel [42], Dask [43], Joblib [44]. Authors in [45] created a distributed framework that uses Ray to manage millions of tasks simultaneously. The proposed framework is claimed to offer programming flexibility, high throughput, and low latencies. In [46], Dispy is used to manage distributing parallel tasks among several computer nodes to decrease the execution time of specific tasks.

Chapter 4

The Proposed Architecture

The proposed architecture is presented in Fig. 4.1. The IDSs located in a set of companies send their updated records, i.e. service logs (1), to a central system (Master) which applies them to multiple models, based on different algorithms. Each model classifies the record as an attack or not an attack and then, the classifications of records per model are sent to a voting system to have the final classification of records and generate a new and classified dataset. The updated datasets are trained to obtain an updated model to be integrated into the IDS of the companies involved (2). All steps are carried out in a regular basis and each company sends a set of log records and receives a new model to improve IDS performance.

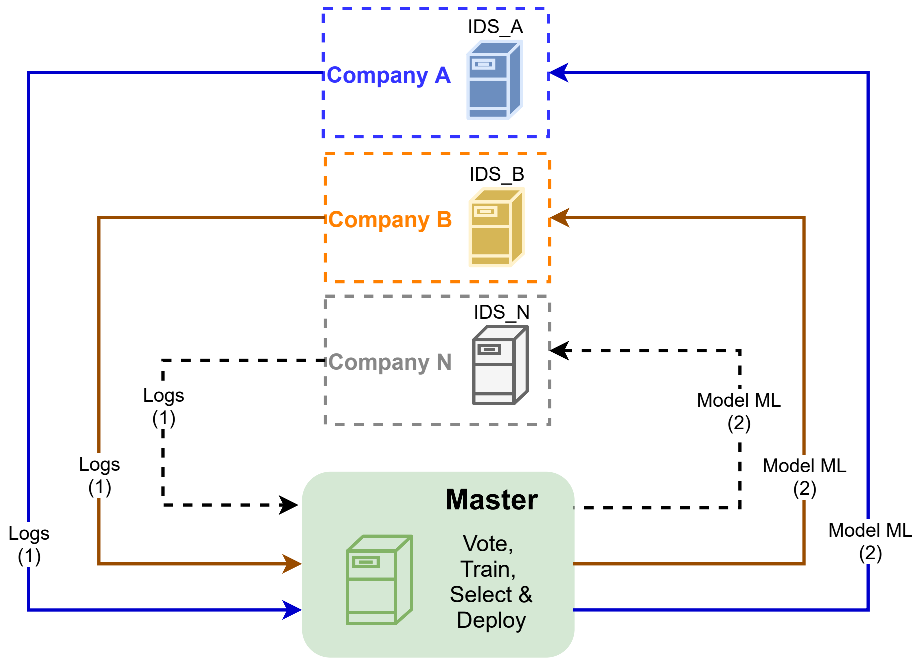


Figure 4.1: Overview of the Proposed Architecture

The detailed operation is presented in Fig. 4.2. The proposed architecture assumes that in a set of companies there is an IDS collecting unlabelled service logs from multiple agents of different service types (e.g. FTP, MySQL, HTTP). These logs are sent in real time for a master system (1). The logs are sent to classifiers (2) that classify the logs (with a binary classification: ”attack” or ”ok”) using different machine learning models. These classified logs are sent to the master (3), where a voting system classifies each record according to the majority. Then, it is created a new dataset categorized by service type, and these new datasets are sent to the trainers (4) to be trained by different algorithms. A new machine learning model is generated for each training (5) and the best model is deployed on the IDS of companies (6).

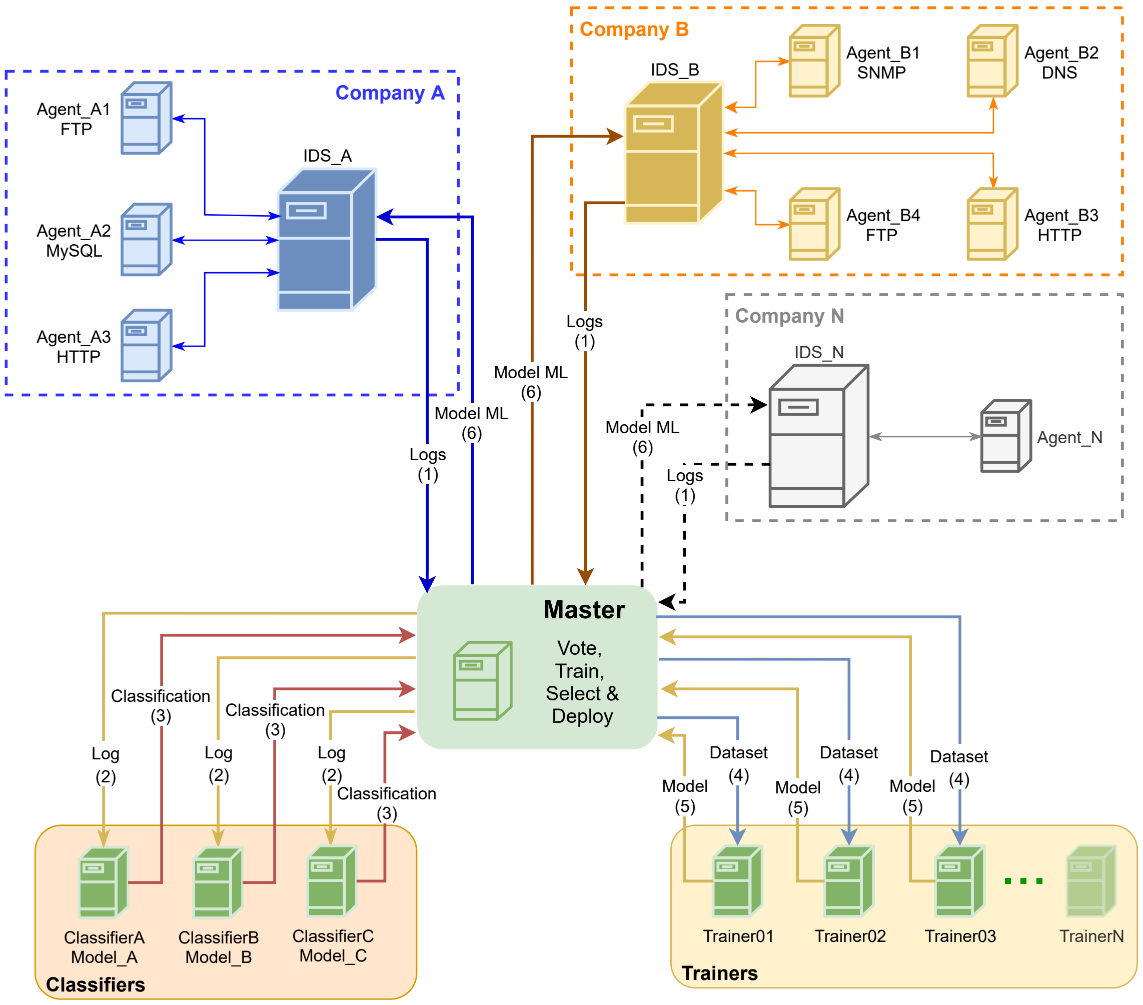


Figure 4.2: Detailed operation of the Proposed Architecture

The operation of the Master assumes a set of classifiers and trainers, whose number can be adjusted according to the need. The records obtained from multiple companies and constant learning will allow the different agents of the companies to improve the performance of detection operation. At same time, the service logs of each company are not known by other companies, which guarantees the privacy and industrial secrecy of their data.

In the first iteration for a given service, the architecture has no previously generated dataset and thus, it comprehends an initial operation with a previously categorized dataset. This initial operation is depicted in Fig. 4.3. An external and categorized dataset (e.g. CICDDoS2019) is sent to the master for the first training (1). This dataset is sent to be trained by different algorithms in Trainers (2). The different machine learning models are generated according to the multiple services. Then, the generated models are deployed into Classifiers (3). The master compares the different models and the one with the best accuracy and precision is sent to the companies' IDS (4).

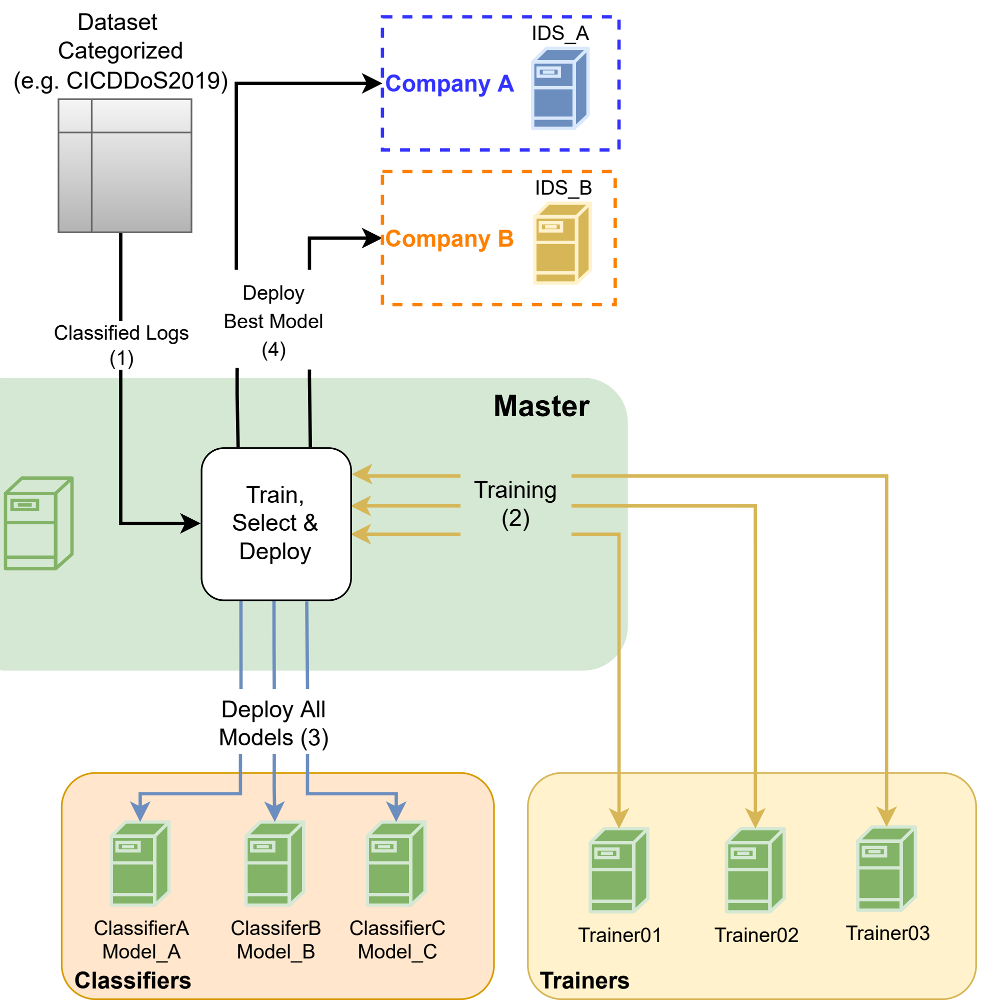
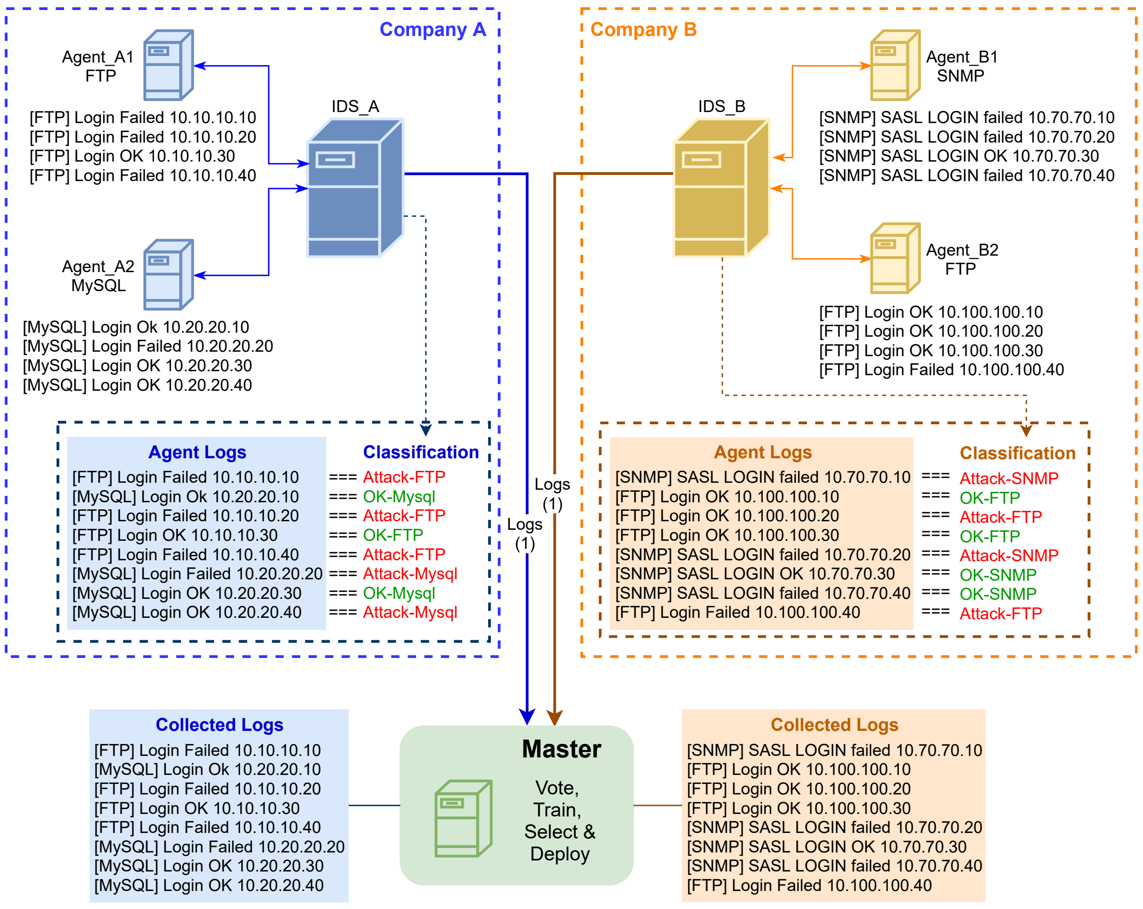


Figure 4.3: Initial operation of the Proposed Architecture

After the first iteration, the proposed architecture is ready to operate in a regular basis. The companies start by collecting the service logs and sending them to the master. Fig. 4.4 details the process of Logs Collection from the IDSs of the companies. In each company, a set of agents produces their service logs; each agent may produce logs for a specific service (e.g. FTP, MySQL, etc). These logs are sent to the IDS of the company, which is composed of multiple anomaly-based IDS using machine learning, with a model for each service type. These models classify each service logs according to the machine learning model in this IDS, and may provide an action against an event (such as to block detected attacks). At the same time, an unclassified version of these logs is sent to the master to generate an enriched dataset and provide an updated model.



```
[FTP] Login Failed 10.10.10.10
[FTP] Login Failed 10.10.10.20
[FTP] Login OK 10.10.10.30
[FTP] Login Failed 10.10.10.40
```

```
[MySQL] Login Ok 10.20.20.10
[MySQL] Login Failed 10.20.20.20
[MySQL] Login OK 10.20.20.30
[MySQL] Login OK 10.20.20.40
```

Agent Logs		Classification
[FTP] Login Failed 10.10.10.10	===	Attack-FTP
[MySQL] Login Ok 10.20.20.10	===	OK-Mysql
[FTP] Login Failed 10.10.10.20	===	Attack-FTP
[FTP] Login OK 10.10.10.30	===	OK-FTP
[FTP] Login Failed 10.10.10.40	===	Attack-FTP
[MySQL] Login Failed 10.20.20.20	===	Attack-Mysql
[MySQL] Login OK 10.20.20.30	===	OK-Mysql
[MySQL] Login OK 10.20.20.40	===	Attack-Mysql

```
[SNMP] SASL LOGIN failed 10.70.70.10
[SNMP] SASL LOGIN failed 10.70.70.20
[SNMP] SASL LOGIN OK 10.70.70.30
[SNMP] SASL LOGIN failed 10.70.70.40
```

```
[FTP] Login OK 10.100.100.10
[FTP] Login OK 10.100.100.20
[FTP] Login OK 10.100.100.30
[FTP] Login Failed 10.100.100.40
```

Agent Logs		Classification
[SNMP] SASL LOGIN failed 10.70.70.10	===	Attack-SNMP
[FTP] Login OK 10.100.100.10	===	OK-FTP
[FTP] Login OK 10.100.100.20	===	Attack-FTP
[FTP] Login OK 10.100.100.30	===	OK-FTP
[SNMP] SASL LOGIN failed 10.70.70.20	===	Attack-SNMP
[SNMP] SASL LOGIN OK 10.70.70.30	===	OK-SNMP
[SNMP] SASL LOGIN failed 10.70.70.40	===	OK-SNMP
[FTP] Login Failed 10.100.100.40	===	Attack-FTP

Collected Logs

```
[FTP] Login Failed 10.10.10.10
[MySQL] Login Ok 10.20.20.10
[FTP] Login Failed 10.10.10.20
[FTP] Login OK 10.10.10.30
[FTP] Login Failed 10.10.10.40
[MySQL] Login Failed 10.20.20.20
[MySQL] Login OK 10.20.20.30
[MySQL] Login OK 10.20.20.40
```

Collected Logs

```
[SNMP] SASL LOGIN failed 10.70.70.10
[FTP] Login OK 10.100.100.10
[FTP] Login OK 10.100.100.20
[FTP] Login OK 10.100.100.30
[SNMP] SASL LOGIN failed 10.70.70.20
[SNMP] SASL LOGIN OK 10.70.70.30
[SNMP] SASL LOGIN failed 10.70.70.40
[FTP] Login Failed 10.100.100.40
```

Figure 4.4: Unclassified logs collection from companies IDSs

These unclassified logs are then received by the Master, and its internal procedures are depicted in Fig. 4.5. When the master receives the unclassified logs from the IDS of the companies (1), it uses a Classification Orchestrator to distribute the service logs by the Classifiers (2) (i.e. Classifier A, B, etc). Each Classifier has the different machine learning models generated in the previous training for each service. The received logs are categorized by the different models of the respective service, where each model sends the classified log to the Majority Voting System (3). With an odd number of classifications, the Majority Voting System outputs the logs with a classification voted by the majority (4). The winning classified logs are sent to a Dataset Assembler, generating a dataset categorized by service. When the dataset reaches a defined limit (e.g. more than 100 thousand classified records), the Training Orchestrator sends the generated dataset to a set of Trainers (5) (i.e. Trainer 1, 2, etc). In each Trainer, the dataset is trained using a different algorithm generating the respective models (6), which are then sent to the Collector. The Collector replaces the models in the Classifiers with the new generated models (7) and compares the models to send to the IDS of companies the one with the best accuracy and precision (8).

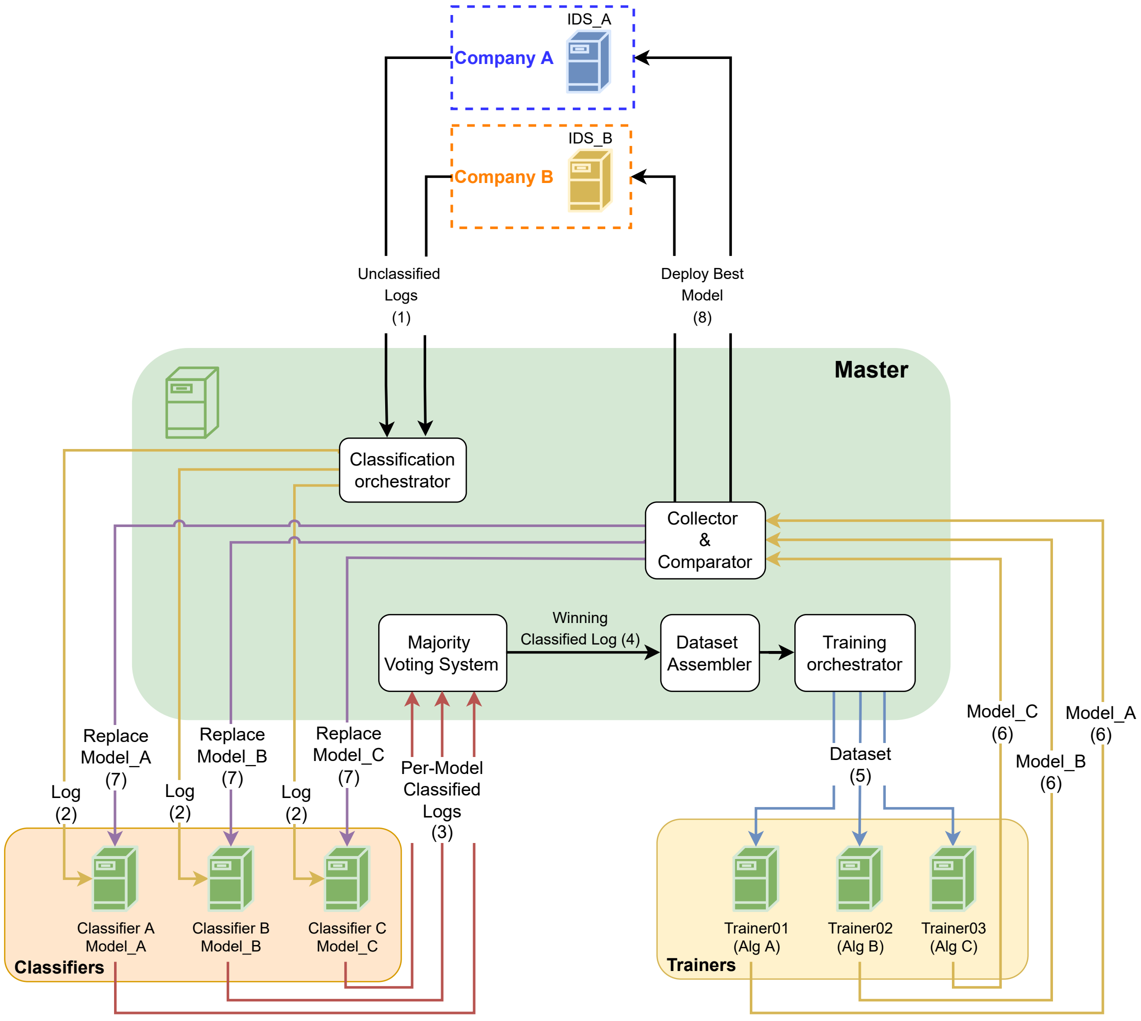
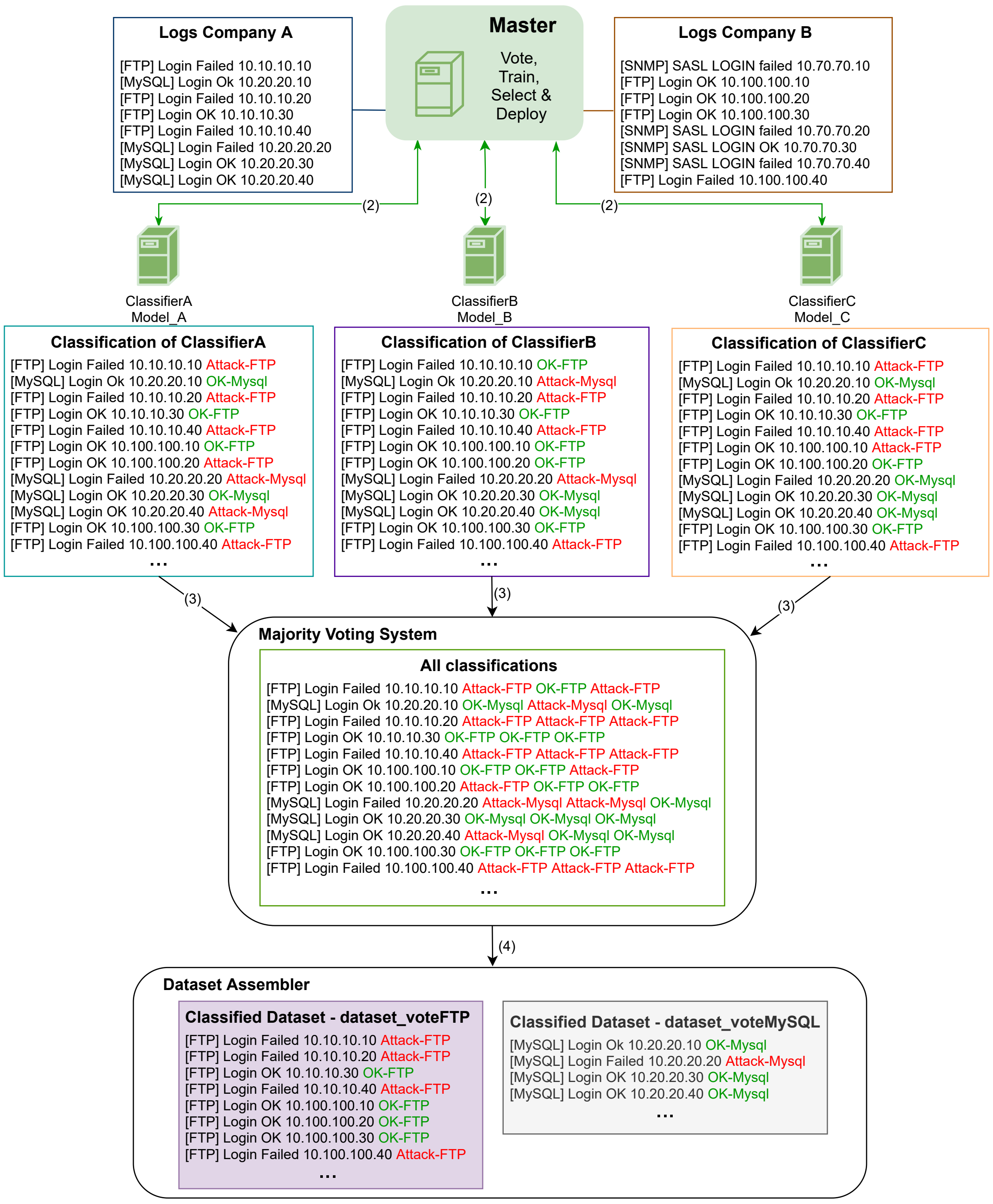


Figure 4.5: Detail of the Master's internal procedures

The Master's internal procedures for classification, voting and dataset generation are depicted in Fig. 4.6. The Master sends the multiple logs received from the companies to the Classifiers (2) to be categorized by different algorithms. The machine learning models classify the logs based on the training and the respective classification of each model is sent to the Majority Voting System (3), that will output these logs classified by the majority. Then, the winning logs are sent (4) to a Data Assembler, where a classified dataset with recent logs is generated. After generating the Dataset, the master requests a new training of the machine learning models (5) with the generated dataset. The models are then sent to the Collector and Comparator (6), and it replaces models in the Classifiers (7) and deploys the best models (8) in the IDS of the companies.



Logs Company A

```
[FTP] Login Failed 10.10.10.10
[MySQL] Login Ok 10.20.20.10
[FTP] Login Failed 10.10.10.20
[FTP] Login OK 10.10.10.30
[FTP] Login Failed 10.10.10.40
[MySQL] Login Failed 10.20.20.20
[MySQL] Login OK 10.20.20.30
[MySQL] Login OK 10.20.20.40
```

Logs Company B

```
[SNMP] SASL LOGIN failed 10.70.70.10
[FTP] Login OK 10.100.100.10
[FTP] Login OK 10.100.100.20
[FTP] Login OK 10.100.100.30
[SNMP] SASL LOGIN failed 10.70.70.20
[SNMP] SASL LOGIN OK 10.70.70.30
[SNMP] SASL LOGIN failed 10.70.70.40
[FTP] Login Failed 10.100.100.40
```

Classification of ClassifierA

```
[FTP] Login Failed 10.10.10.10 Attack-FTP
[MySQL] Login Ok 10.20.20.10 OK-Mysql
[FTP] Login Failed 10.10.10.20 Attack-FTP
[FTP] Login OK 10.10.10.30 OK-FTP
[FTP] Login Failed 10.10.10.40 Attack-FTP
[FTP] Login OK 10.100.100.10 OK-FTP
[FTP] Login OK 10.100.100.20 Attack-FTP
[MySQL] Login Failed 10.20.20.20 Attack-Mysql
[MySQL] Login OK 10.20.20.30 OK-Mysql
[MySQL] Login OK 10.20.20.40 Attack-Mysql
[FTP] Login OK 10.100.100.30 OK-FTP
[FTP] Login Failed 10.100.100.40 Attack-FTP
...
```

Classification of ClassifierB

```
[FTP] Login Failed 10.10.10.10 OK-FTP
[MySQL] Login Ok 10.20.20.10 Attack-Mysql
[FTP] Login Failed 10.10.10.20 Attack-FTP
[FTP] Login OK 10.10.10.30 OK-FTP
[FTP] Login Failed 10.10.10.40 Attack-FTP
[FTP] Login OK 10.100.100.10 OK-FTP
[FTP] Login OK 10.100.100.20 OK-FTP
[MySQL] Login Failed 10.20.20.20 Attack-Mysql
[MySQL] Login OK 10.20.20.30 OK-Mysql
[MySQL] Login OK 10.20.20.40 OK-Mysql
[FTP] Login OK 10.100.100.30 OK-FTP
[FTP] Login Failed 10.100.100.40 Attack-FTP
...
```

Classification of ClassifierC

```
[FTP] Login Failed 10.10.10.10 Attack-FTP
[MySQL] Login Ok 10.20.20.10 OK-Mysql
[FTP] Login Failed 10.10.10.20 Attack-FTP
[FTP] Login OK 10.10.10.30 OK-FTP
[FTP] Login Failed 10.10.10.40 Attack-FTP
[FTP] Login OK 10.100.100.10 Attack-FTP
[FTP] Login OK 10.100.100.20 OK-FTP
[MySQL] Login Failed 10.20.20.20 OK-Mysql
[MySQL] Login OK 10.20.20.30 OK-Mysql
[MySQL] Login OK 10.20.20.40 OK-Mysql
[FTP] Login OK 10.100.100.30 OK-FTP
[FTP] Login Failed 10.100.100.40 Attack-FTP
...
```

Majority Voting System

All classifications

```
[FTP] Login Failed 10.10.10.10 Attack-FTP OK-FTP Attack-FTP
[MySQL] Login Ok 10.20.20.10 OK-Mysql Attack-Mysql OK-Mysql
[FTP] Login Failed 10.10.10.20 Attack-FTP Attack-FTP Attack-FTP
[FTP] Login OK 10.10.10.30 OK-FTP OK-FTP OK-FTP
[FTP] Login Failed 10.10.10.40 Attack-FTP Attack-FTP Attack-FTP
[FTP] Login OK 10.100.100.10 OK-FTP OK-FTP Attack-FTP
[FTP] Login OK 10.100.100.20 Attack-FTP OK-FTP OK-FTP
[MySQL] Login Failed 10.20.20.20 Attack-Mysql Attack-Mysql OK-Mysql
[MySQL] Login OK 10.20.20.30 OK-Mysql OK-Mysql OK-Mysql
[MySQL] Login OK 10.20.20.40 Attack-Mysql OK-Mysql OK-Mysql
[FTP] Login OK 10.100.100.30 OK-FTP OK-FTP OK-FTP
[FTP] Login Failed 10.100.100.40 Attack-FTP Attack-FTP Attack-FTP
...
```

Dataset Assembler

Classified Dataset - dataset_voteFTP

```
[FTP] Login Failed 10.10.10.10 Attack-FTP
[FTP] Login Failed 10.10.10.20 Attack-FTP
[FTP] Login OK 10.10.10.30 OK-FTP
[FTP] Login Failed 10.10.10.40 Attack-FTP
[FTP] Login OK 10.100.100.10 OK-FTP
[FTP] Login OK 10.100.100.20 OK-FTP
[FTP] Login OK 10.100.100.30 OK-FTP
[FTP] Login Failed 10.100.100.40 Attack-FTP
...
```

Classified Dataset - dataset_voteMySQL

```
[MySQL] Login Ok 10.20.20.10 OK-Mysql
[MySQL] Login Failed 10.20.20.20 Attack-Mysql
[MySQL] Login OK 10.20.20.30 OK-Mysql
[MySQL] Login OK 10.20.20.40 OK-Mysql
...
```

Figure 4.6: Master's internal procedures for classification, voting and dataset generation

In particular, the training, selection and deployment stages comprehend 3 phases, depicted in Fig. 4.7. In phase 1 the new classified dataset is trained by different algorithms in the Trainers to generate new machine learning models. In phase 2, the new models are deployed in the Classifiers. The new machine learning models will be used in the voting system that classifies the most recent records received from the companies as shown previously in Fig. 4.6. Finally, in phase 3, the model with better accuracy and precision is sent to the IDS of companies, to increase their performance in detecting attacks.

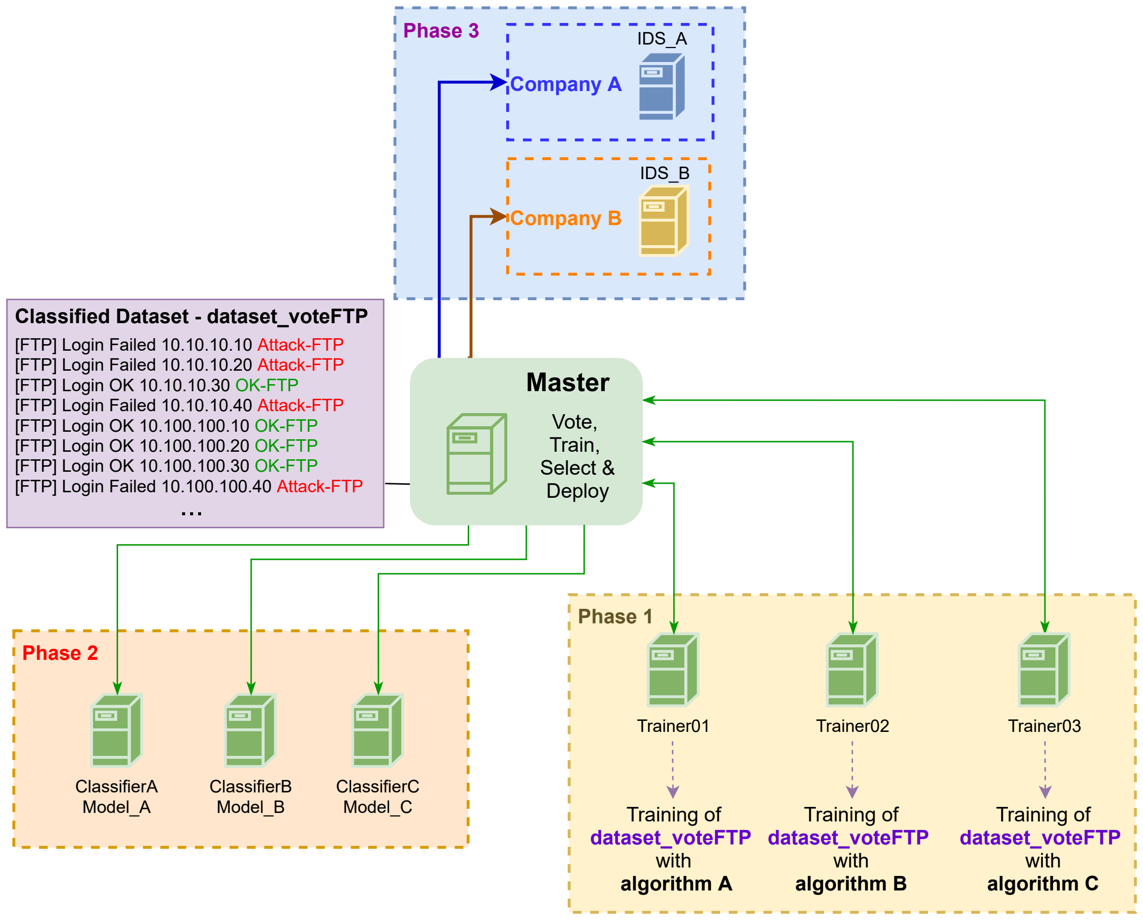

Phase 3
IDS_A
Company A
IDS_B
Company B
Classified Dataset - dataset_voteFTP
[FTP] Login Failed 10.10.10.10 Attack-FTP
[FTP] Login Failed 10.10.10.20 Attack-FTP
[FTP] Login OK 10.10.10.30 OK-FTP
[FTP] Login Failed 10.10.10.40 Attack-FTP
[FTP] Login OK 10.100.100.10 OK-FTP
[FTP] Login OK 10.100.100.20 OK-FTP
[FTP] Login OK 10.100.100.30 OK-FTP
[FTP] Login Failed 10.100.100.40 Attack-FTP
...
Master
Vote, Train, Select & Deploy
Phase 2
ClassifierA Model_A
ClassifierB Model_B
ClassifierC Model_C
Phase 1
Trainer01
Trainer02
Trainer03
Training of dataset_voteFTP with algorithm A
Training of dataset_voteFTP with algorithm B
Training of dataset_voteFTP with algorithm C

Figure 4.7: Training, selection and deployment phases

The proposed architecture assumes the regular training of different machine learning models, which is a time and resource consuming task. In order to be implemented, scalable testbeds and parallel processing tools can be used.

Chapter 5

Validation

This chapter details the validation stage of the proposed architecture. The validation stage starts by selecting and preparing a dataset. Then, tests were performed to select the three best algorithms to be implement in the proposed architecture. Finally, the performance of the proposed architecture was tested with four iterations. These stages are detailed in the following sections.

5.1 Dataset Selection and Preparation

The Supervised learning algorithms need categorized datasets. The works developed that add supervised learning algorithms to a IDS mainly use three datasets: KDD'99 [47], NSL-KDD [48], UNSW-NB15 [49], and CICDDoS2019 dataset [50]. These datasets were built with records from 1999, 2009, 2015, and 2019, respectively. With the constant evolution of attacks, for efficient protection against them, it is essential to use datasets with real and recent records and thus, the CICDDoS2019, a dataset provided by the Canadian Institute for Cybersecurity, University of New Brunswick, Canada, was selected and analysed in detail to validate the proposed architecture.

The CICDDoS2019 dataset features 50 million records from 2019, distributed by 14 different labels, as presented in Table 5.1. The records in this dataset relate to benign traffic and recent DDoS attacks on various services (NTP, LDAP, SSDP, FTP, MSSQL, DNS, among others). The main focus of this thesis is to create an architecture where IDSs are efficient in protecting their agents. Thus, the use of this dataset allows machine learning algorithms to train with records of attacks on various services, increasing the knowledge of the model for situations similar to those present in the dataset.

Table 5.1: Description of each label and number of instances

Attribute (Class Label)	**Description**	**Instances Number**
DDoS_WebDDoS	Send multiple requests to the attacked web resource with the aim of exceeding the website's resource [51]	439
Benign	Legitimate traffic	56 863
DDoS_Portmap	An attack on TCP or UDP port 111 which is a service used to direct clients to the proper port number so they can communicate with the requested Remote Procedure Call (RPC) service [52]	186 960
DDoS_UDP-Lag	Attempts to break the client-server connection [53]	366 461
DDoS_NTP	Pool of unidentified UDP network packets sent to NTP servers [54]	1 202 642
DDoS_SYN	SYN flood attack is to consume all resources by sending huge amount of data or a large number of SYN requests packets [55]	1 582 289
DDoS_LDAP	Exploit web-based applications that construct LDAP statements based on user inputs [56]	2 179 930
DDoS_SSDP	An attack that an enormous amount of traffic are generated to exploiting the UPnP protocol [57]	2 610 611
DDoS_UDP	UDP packets are sent to a victim to slow down or go down its resources [58]	3 134 645
DDoS_NetBIOS	An attack that uses the NetBIOS service [59]	4 093 279
DDoS_MSSQL	Execute malicious SQL statements [60]	4 522 492
DDoS_DNS	Exploit the vulnerabilities in the DNS [61]	5 071 011
DDoS_SNMP	Sends a large number of requests with spoofed IP address to several devices [62]	5 159 870
DDoS_TFTP	A amplification DDoS attack based on TFTP [63]	20 082 580

The proposed architecture was evaluated in a set of validation tests, using the CICDDoS2019 dataset. To prepare this dataset to be used, a feature selection was made based on the most important features. This selection used the Extra Trees Classifier class from scikit-learn as presented in Listing 5.1.

```
{
    from sklearn.ensemble import ExtraTreesClassifier
    import numpy as np

    df = pd.read_csv(dataset)
    y = df['Label']
    x = df.drop('Label', axis=1)

    extra_tree_forest = ExtraTreesClassifier(n_estimators=1000,
        criterion='entropy', max_features='auto')
    extra_tree_forest.fit(x, y)

    feature_importance = extra_tree_forest.feature_importances_
    feature_importance_normalized = np.std([tree.feature_importances_
        for tree in extra_tree_forest.estimators_], axis=0)

    index = np.argsort(feature_importance)[::-1]
}
```

Listing 5.1: Script for the Selection of relevant features

As result, the features with importance greater than 2.5% were considered and selected. Table 5.2 presents a brief description of each of 23 selected features.

Table 5.2: Description of each select feature

Feature	**Description**
Source IP	IP address from which packets were sent
Source Port	Port from which packets were sent
Destination IP	Address IP to receive packets
Protocol	Protocol for data transmission
Timestamp	Time of transmission
Fwd Packet Length Max	Maximum size of packet in forward direction
Fwd Packet Length Min	Minimum size of packet in forward direction
Fwd Packet Length Mean	Mean size of packet in forward direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Flow Bytes/s	Number of flow bytes per second
Flow Packets/s	Minimum length of a packet
Min Packet Length	Minimum length of a packet
Max Packet Length	Maximum length of a packet
Packet Length Mean	Mean length of a packet
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG
CWE Flag Count	Number of packets with CWE
Down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Avg Fwd Segment Size	Average size observed in the forward direction
Avg Bwd Segment Size	Average number of bytes bulk rate in the forward direction
Init_Wing_bytes_forward	The total number of bytes sent in initial window in the forward direction
Inbound	Traffic originates from outside the network

5.2 Machine Learning Algorithms Selection

After selecting the features with most importance, a selection of the machine algorithms that will train the dataset, in the trainers, was performed. The goal is to have an odd-numbered set of trainers each with a machine learning algorithm and, for the current validation tests, three algorithms needed to be selected. The best three algorithms were selected from the following range: Decision Tree Classifier (DTC), Random Forest Classifier (RFC), K-Nearest Neighbors (KNN), Simple Logistics (SLog), and Support Vector Machine (SVM).

These five algorithms were compared in terms of accuracy, precision and execution time when using the following subsets of records from the CICDDoS2019 dataset:

- Subset A - collected using 191 694 records (186 960 DDoS_Portmap and 4734 Benign);
- Subset B - collected using 300 661 records (287 873 DDoS_NTP and 12 788 Benign);
- Subset C - collected using 395 572 records (395 415 DDoS_Syn and 157 Benign);
- Subset D - collected using 544 983 records (544 647 DDoS_LDAP and 336 Benign);
- Subset E - collected using 652 652 records (652 394 DDoS_SSDP and 258 Benign);
- Subset F - collected using 783 661 records (782 590 DDoS_UDP and 1071 Benign);
- Subset G - collected using 1 023 320 records (1 022 456 DDoS_NetBIOS and 864 Benign);
- Subset H - collected using 1 130 623 records (1 129 604 DDoS_MSSQL and 1019 Benign);
- Subset I - collected using 1 267 750 records (1 265 727 DDoS_DNS and 2023 Benign);
- Subset J - collected using 1 298 968 records (1 289 043 records DDoS_SNMP and 925 Benign);
- Subset K - collected using 3 012 387 records (3 010 540 DDoS_TFTP and 1847 Benign);

- Subset Z - collected using 1 059 153 records (125 032 DDoS_DNS, 54 490 DDoS_LDAP, 112 410 DDoS_MSSQL, 17 740 DDoS_NTP, 128 951 DDoS_SNMP, 65 155 DDoS_SSDP, 300 367 DDoS_TFTP, 77 466 DDoS_UDP, 14 280 Portmap, 101 756 DDoS_NetBIOS and 21 955 Benign). Although this subset contains records from several datasets and therefore has multi classification, it was adjusted to also have binary classification (Attack, Benign).

Table 5.3 presents the accuracy obtained by each algorithm for each subset. From the results obtained, it can be verified that the subsets with records by service (subsets A to K) presented an accuracy ranging from 90% to 96%. In turn, the results of subset Z (records of multi services) presented a lower accuracy ranging from 79% to 81%. The accuracy results show that the best three algorithms are the Simple Logistics, the Decision Tree Classifier, and the Random Forest Classifier.

Table 5.3: Results of the accuracy of each algorithm for each subset

	Accuracy (%)				
	SLog	**DTC**	**RFC**	**KNN**	**SVM**
Subset A	93.78	94.78	93.32	92.61	92.15
Subset B	94.48	91.73	92.54	92.31	92.08
Subset C	92.53	92.39	92.28	90.67	91.89
Subset D	92.28	93.88	92.05	91.16	90.12
Subset E	94.24	93.29	93.85	92.64	93.03
Subset F	93.23	95.74	94.63	92.39	91.75
Subset G	93.28	91.69	92.42	90.33	90.77
Subset H	92.39	92.16	91.89	91.04	90.35
Subset I	92.57	91.31	92.53	90.71	90.14
Subset J	94.32	94.86	95.23	93.45	92.79
Subset K	94.03	93.29	93.14	92.04	91.88
Subset Z	80.93	80.56	81.26	79.27	80.19
Average	92.34	92.14	92.10	90.72	90.60

Table 5.4 presents the precision for each algorithm for each subset. From the results obtained, it can be verified that the subsets with records by service (subsets A to K) presented a precision ranging from 88% to 93%. In turn, the results of subset Z (records of multi services) presented a precision ranging from 80% to 82%. The precision results show that the best three algorithms are the Random Forest Classifier, the Decision Tree Classifier, and the Simple Logistics.

Table 5.4: Results of the precision of each subset for each tested algorithm

	Precision (%)				
	RFC	**DTC**	**SLog**	**KNN**	**SVM**
Subset A	92.04	91.89	91.72	90.38	90.14
Subset B	91.83	92.04	91.23	90.01	89.88
Subset C	92.36	91.99	91.00	89.35	89.83
Subset D	92.28	91.82	91.23	90.04	90.32
Subset E	91.76	91.35	91.28	88.36	89.63
Subset F	92.48	92.36	92.01	90.13	89.32
Subset G	91.82	91.75	90.83	89.33	88.64
Subset H	92.86	91.38	91.14	90.37	89.82
Subset I	91.42	90.53	90.86	88.63	88.93
Subset J	92.43	91.83	91.07	89.77	90.04
Subset K	92.83	92.12	91.78	90.42	90.00
Subset Z	81.75	80.54	80.35	79.79	80.04
Average	91.32	90.80	90.38	88.88	88.88

From the results presented in Table 5.3 and Table 5.4, it is observed that the highest values regarding accuracy and precision were obtained for the datasets with records per service (Subset A to K).

To balance the subsets and test the algorithms with balanced subsets oversampling techniques were applied. For this, the imbalanced-learn library from scikit-learn was chosen, using the Synthetic Minority Oversampling Technique (SMOTE) method with the minority argument to resample only the minority class and with the seed at 7. SMOTE is a statistical technique for increasing the number of instances in a dataset such that all class labels have the same number of instances, as presented in Listing 5.2.

```
{
    from imblearn.over_sampling import SMOTE

    sm = SMOTE(sampling_strategy='minority', random_state=7)
}
```

Listing 5.2: Applying SMOTE to balance the dataset

The five algorithms previously mentioned were compared in terms of accuracy, precision and execution time using the new subsets:

- Subset A_Over - Oversampling over minority class was applied to Subset A and result a balanced dataset with 373 920 records (186 960 DDoS_Portmap and 186 960 Benign);
- Subset B_Over - Oversampling over minority class was applied to Subset B and result a balanced dataset with 575 746 records (287 873 DDoS_NTP and 287 873 Benign);
- Subset C_Over - Oversampling over minority class was applied to Subset C and result a balanced dataset with 790 83 records (395 415 DDoS_Syn and 395 415 Benign);
- Subset D_Over - Oversampling over minority class was applied to Subset D and result a balanced dataset with 1 089 294 records (544 647 DDoS_LDAP and 544 647 Benign);
- Subset E_Over - Oversampling over minority class was applied to Subset E and result a balanced dataset with 1 304 788 records (652 394 DDoS_SSDP and 652 394 Benign);
- Subset F_Over - Oversampling over minority class was applied to Subset F and result a balanced dataset with 1 565 180 records (782 590 DDoS_UDP and 782 590 Benign);
- Subset G_Over - Oversampling over minority class was applied to Subset G and result a balanced dataset with 2 044 912 records (1 022 456 DDoS_NetBIOS and 1 022 456 Benign);

- Subset H_Over - Oversampling over minority class was applied to Subset H and result a balanced dataset with 2 259 208 records (1 129 604 DDoS_MSSQL and 1 129 604 Benign);
- Subset I_Over - Oversampling over minority class was applied to Subset I and result a balanced dataset with 2 531 454 records (1 265 727 DDoS_DNS and 1 265 727 Benign);
- Subset J_Over - Oversampling over minority class was applied to Subset J and result a balanced dataset with 2 578 086 records (1 289 043 DDoS_SNMP and 1 289 043 Benign);
- Subset K_Over - Oversampling over minority class was applied to Subset K and result a balanced dataset with 6 021 080 records (3 010 540 DDoS_TFTP and 3 010 540 Benign);
- Subset Z_Over - Oversampling over minority class was applied to Subset Z and result a balanced dataset with 2 074 396 records (1 037 198 Attack and 1 037 198 Benign);

Table 5.5 presents the accuracy in percentage obtained by each algorithm for each subset after applying oversampling. From the results obtained, it can be verified that the subsets with records by service (subsets A_Over to K_Over) presented an accuracy ranging from 89% to 94%. In turn, the results of subset Z_Over (records of multi services) presented a lower accuracy ranging from 78% to 82%. The accuracy results show that the best three algorithms are the Random Forest Classifier, the Simple Logistics, and the Decision Tree Classifier.

Table 5.5: Results of the accuracy of each algorithm for each subset after applying over-sampling

	Accuracy (%)				
	RFC	**SLog**	**DTC**	**KNN**	**SVM**
Subset A_Over	93.25	92.38	91.48	91.87	88.83
Subset B_Over	92.33	92.04	91.75	90.03	89.49
Subset C_Over	91.37	91.79	91.88	89.95	88.32
Subset D_Over	92.85	90.57	92.43	90.99	89.43
Subset E_Over	94.08	93.08	91.88	91.23	90.34
Subset F_Over	91.54	91.16	91.22	90.91	86.12
Subset G_Over	91.28	90.86	91.31	90.04	88.33
Subset H_Over	92.38	91.32	91.07	89.58	87.26
Subset I_Over	91.33	90.52	90.95	88.78	86.77
Subset J_Over	92.93	91.27	92.51	92.57	91.34
Subset K_Over	91.80	91.73	90.01	89.52	88.36
Subset Z_Over	81.73	80.78	80.28	78.68	79.32
Average	91.41	90.63	90.56	89.51	87.83

Table 5.6 presents the precision in percentage for each algorithm for each subset. From the results obtained, it can be verified that the subsets with records by service (subsets A_Over to K_Over) presented a precision ranging from 88% to 93%. In turn, the results of subset Z_Over (records of multi services) presented a precision ranging from 78% to 81%. The precision results show that the best three algorithms are the Random Forest Classifier, the Decision Tree Classifier, and the Simple Logistics.

Table 5.6: Results of the precision of each subset after applying oversampling for each tested algorithm

	Precision (%)				
	RFC	**DTC**	**SLog**	**KNN**	**SVM**
Subset A_Over	91.96	90.31	91.36	90.93	90.19
Subset B_Over	91.04	91.22	90.12	89.82	89.06
Subset C_Over	91.43	91.10	90.36	90.17	89.36
Subset D_Over	92.01	91.72	91.27	90.34	90.48
Subset E_Over	91.19	91.28	90.99	89.43	88.56
Subset F_Over	91.46	91.88	91.45	89.52	88.04
Subset G_Over	90.39	90.31	89.88	90.01	89.31
Subset H_Over	91.24	90.87	90.33	90.01	88.89
Subset I_Over	90.15	91.32	90.85	89.32	87.23
Subset J_Over	92.61	92.83	90.78	91.82	90.71
Subset K_Over	92.31	90.53	91.82	87.85	89.63
Subset Z_Over	80.93	79.81	80.01	78.13	78.99
Average	90.56	90.27	89.94	88.95	88.37

From the results presented in Table 5.5 and Table 5.6, it is observed that the highest values regarding accuracy and precision were obtained for the datasets with records per service (Subset A_Over to K_Over). As verified, both in balanced and unbalanced datasets the best results were obtained in datasets with records per service.

There was a part of the subset (30%) that was not trained to later test the model created. In the end, the model obtained the same accuracy values using the training dataset part and the test dataset part, demonstrating that there was no overfitting. Underfitting was resolved naturally by using the most important features.

Table 5.7 presents the elapsed runtime for each algorithm and per subset. Table 5.8 presents the elapsed runtime for each algorithm and per subset, after applying oversampling. To obtain these results, each algorithm and subset was trained using a workstation featuring an Intel® Core^{TM} i5-9400 CPU @ 2.90GHz, with 32 GB of RAM. From the results obtained, the Decision Tree Classifier, the Simple Logistics, and the Random Forest

Classifier algorithms trained in a shorter time, in average, than the K-Nearest Neighbors and Support Vector Machine algorithms.

Table 5.7: Elapsed runtime results for each algorithm and per subset

	Elapsed Runtime (s)				
	DTC	**SLog**	**RFC**	**KNN**	**SVM**
Subset A	1	98	40	97	57
Subset B	1	78	82	431	80
Subset C	1	132	87	270	91
Subset D	1	36	79	816	120
Subset E	2	154	174	1199	170
Subset F	2	211	219	1668	234
Subset G	2	160	205	2779	6721
Subset H	2	416	262	3603	115
Subset I	3	206	358	3862	11278
Subset J	3	199	371	4034	965
Subset K	4	314	935	9208	11591
Subset Z	4	238	472	2988	1834
Average	2.17	186.83	273.67	2579.58	2771.33

Table 5.8: Elapsed runtime results for each algorithm and per subset after applying oversampling

	Elapsed Runtime (s)				
	DTC	**SLog**	**RFC**	**KNN**	**SVM**
Subset A_Over	2	124	161	426	1034
Subset B_Over	2	4	361	1600	7043
Subset C_Over	3	89	257	927	1764
Subset D_Over	3	14	381	3258	7485
Subset E_Over	4	12	595	4397	3749
Subset F_Over	5	37	782	6333	17919
Subset G_Over	8	25	1060	11754	43054
Subset H_Over	14	15	1236	12978	29151
Subset I_Over	14	25	1347	16983	26343
Subset J_Over	16	18	1489	19069	47896
Subset K_Over	30	46	2276	99969	86559
Subset Z_Over	12	32	1128	12004	35127
Average	9.42	36.75	922.75	15808.17	25593.67

Regarding the accuracy, the precision, and the elapsed runtime obtained in the tests, the three best algorithms were the Decision Tree Classifier, the Random Forest Classifier, and the Simple Logistics.

5.3 Performance Results and Analysis

In order to test the performance of the proposed architecture, their components (the Master, the Classifiers, and the Trainers) were implemented in a workstation featuring an Intel® Core^{TM} i5-9400 CPU @2.90GHz, with 32 GB of RAM. From the preliminary tests regarding models accuracy, precision, and elapsed runtime, the Decision Tree Classifier, the Random Forest Classifier, and the Simple Logistics were selected to be deployed in the Classifiers. The datasets are generated per service, as they present higher results for accuracy and precision, and they are generated in the dataset assembler each 100 thousand classified records.

Five samples of the main CICDDoS2019 dataset were prepared and tested in two scenarios: ”baseline” and ”proposed architecture”. These samples represent consecutive iterations for the proposed architecture and they were obtained from the ”DDoS_TFTP” service. Each sample, i.e. Sample #0 to Sample #4, includes 200k randomly records sequential in time, i.e. if the first sample was collected from 14h to 16h, the following is from 16h to 20h.

The Sample #0 is used for the initial operation of the proposed architecture, and it was trained in the trainers using the three algorithms selected, generating three machine learning models. The accuracy and precision results for Sample #0 using the 3 models are presented in Table 5.9. The results show that the Decision Tree Classifier was the best regarding accuracy and precision and thus, it was selected as the model to evaluate the next sample, i.e. Sample #1, in both scenarios.

Table 5.9: Accuracy and Precision for Sample #0

	Sample #0	
Algorithm	**Accuracy (%)**	**Precision (%)**
Decision Tree Classifier	93.04	91.85
Random Forest Classifier	92.43	91.07
Simple Logistics	91.78	90.32

In the baseline scenario, the Decision Tree Classifier model was deployed directly in an IDS and tested with the remaining four different categorized samples, to obtain their accuracy and precision. In the proposed architecture scenario, the remaining procedures are depicted in Fig. 5.1. The three models were deployed in the Classifiers, and the Decision Tree Classifier model was loaded into the IDS. This model was tested with Sample #1 to calculate accuracy and precision (1). Each sample was sent, unclassified, to the master (2) and the Classifiers categorize each sample, and these samples were classified by the majority voting system (3). The classified samples were sent to Trainers (4) and each model was deployed in the classifiers and the best deployed in the IDS (5). The process was repeated for all remaining samples, and the Classifiers had the models generated by the training of the previous sample. In the execution of the tests presented above, for each sample tested in each architecture, the accuracy and precision of the model were obtained to compare the results.

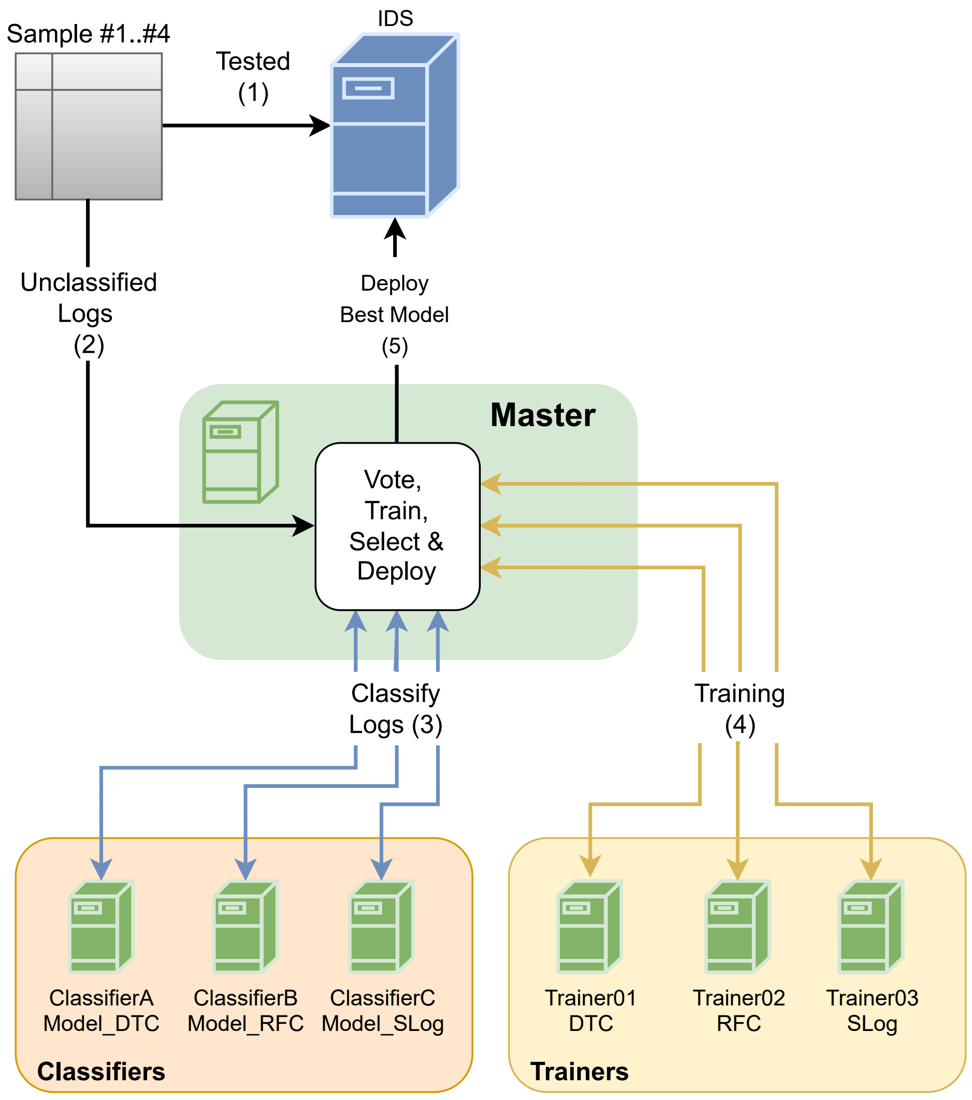


Figure 5.1: Proposed Architecture validation scenario

Table 5.10 presents the models used for each sample for the Baseline and for the Proposed Architecture. In the Baseline there are no new trainings, so the model used was always the result of the training of Sample #0, that is, Decision Tree Classifier. In the Proposed Architecture, with each new training, the best model is sent to the IDS, and thus, the Decision Tree Classifier was selected for Sample #1, the Random Forest Classifier was selected for Sample #2, and the Decision Tree Classifier was selected for Samples #3 and #4.

Table 5.10: Models used for the Baseline and the Proposed Architecture scenarios

	Baseline	**Proposed Architecture**
Sample #1	Decision Tree Classifier	Decision Tree Classifier
Sample #2		Random Forest Classifier
Sample #3		Decision Tree Classifier
Sample #4		Decision Tree Classifier

Table 5.11 and Table 5.12 presents, respectively, the accuracy and the precision values for each sample tested, and their difference in percentage points (p.p.) and percentage, in the Baseline and in the Proposed Architecture scenarios.

Table 5.11: Accuracy results of Baseline and Proposed Architecture scenarios

	Baseline (B) (%)	**Proposed Architecture (PA) (%)**	**PA-B (p.p.)**	**PA-B (%)**
Sample #1	87.85	87.85	0	0
Sample #2	87.17	93.27	+ 6.10	+ 6.99
Sample #3	88.81	94.89	+ 6.08	+ 6.85
Sample #4	85.29	95.11	+ 9.82	+ 11.51

Table 5.12: Precision results of Baseline and Proposed Architecture scenarios

	Baseline (B) (%)	**Proposed Architecture (PA) (%)**	**PA-B (p.p.)**	**PA-B (%)**
Sample #1	91.76	91.76	0	0
Sample #2	89.49	95.39	+ 5.90	+ 6.59
Sample #3	90.22	95.77	+ 5.55	+ 6.15
Sample #4	86.27	95.94	+ 9.67	+ 11.21

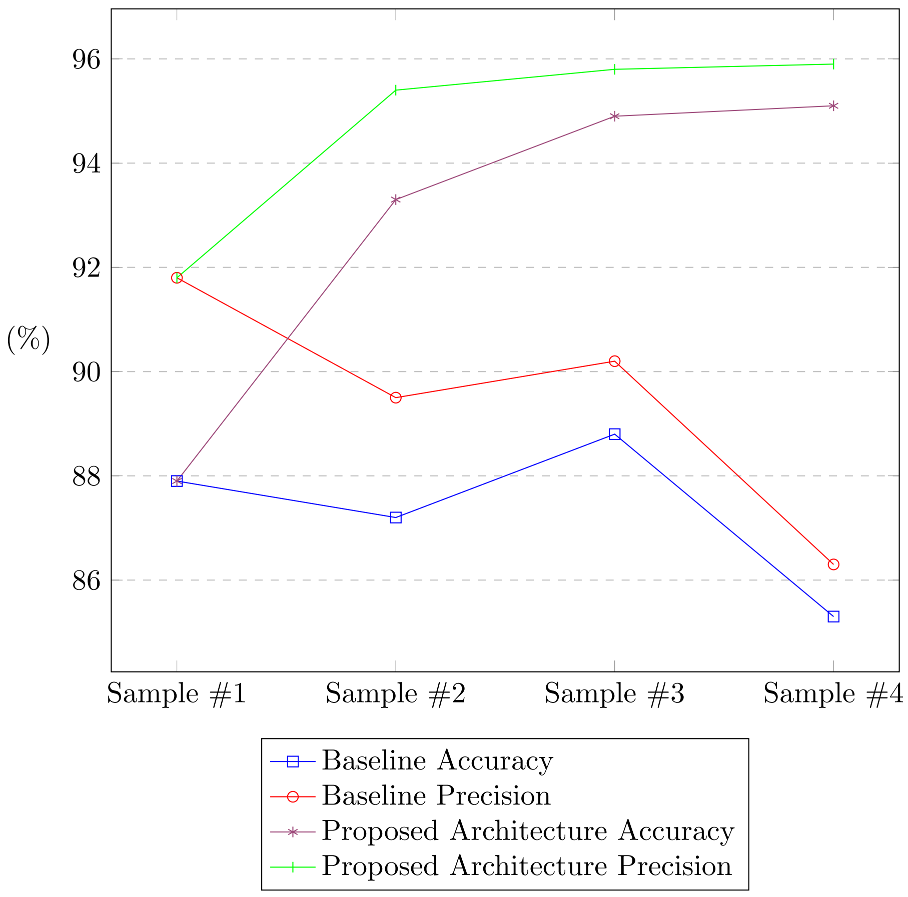


Figure 5.2: Results of accuracy and precision for the Baseline and the Proposed Architecture scenarios

Fig. 5.2 draws the results of Table 5.11 and Table 5.12. From these results, it can be verified that in the proposed architecture, as the model is always trained with a dataset that was categorized with the contribution of several algorithms, both the accuracy and the precision improve over the time. After four consecutive iterations in the proposed architecture, the accuracy increased by 9.82 p.p. or 11.51%, and the precision increased by 9.67 p.p. or 11.21%, when compared to the baseline scenario.

The implementation of the proposed architecture assumes recurrent and intensive classification and training tasks, requiring time and resources. In order to expediently run these tasks and apply the best model in each company, the proposed architecture was deployed in Fed4FIRE+, a federated testbed, with Ray to distribute the tasks by the available resources, as presented in Fig 5.3. Other federated testbed platforms and distributed tasks tools could be used, in case they present similar specifications.

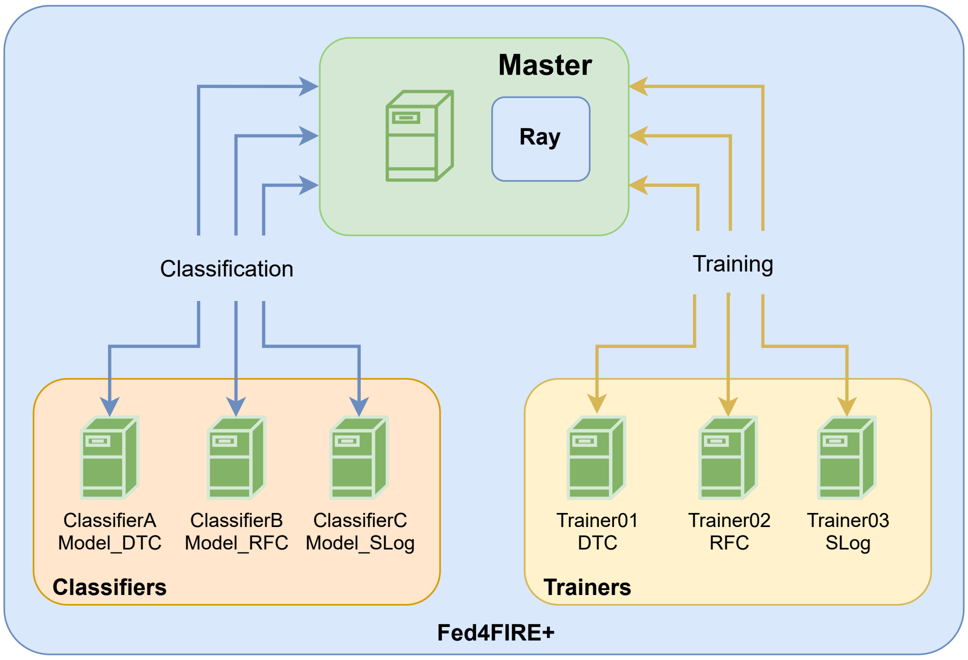


Figure 5.3: Implementation of the Proposed Architecture using Fed4FIRE+ and Ray

A set of tests was carried out to verify the runtime differences, using Fed4FIRE+ with Ray, on a single server and on a multiple servers, in each step of iteration (classification and training a subset). The single server used a workstation featuring an Intel® Core™ i5-9400 CPU @ 2.90GHz, with 32 GB of RAM, and the multiple servers used one workstation (with the same specifications of the single server) for each classification or training algorithm task.

Table 5.13, Table 5.14, Table 5.15, and Table 5.16 presents the time taken to classify 200k records on a single server (without ray) and multiple servers (distributed tasks managed by ray), in four samples or iterations, i.e. Sample #1, Sample #2, Sample #3, Sample #4. From the results obtained, it can be verified that, as in a single server (Server_0), the classifications in the different models are carried out in sequence, the total time represents the sum of each stage, between 7 010 ms (Sample #1) and 7 068 ms (Sample #3). In turn, with multiple servers (Server_1, Server_2, and Server_3) the tasks run in parallel, the total time is equal to the time taken by the longest task, so between 4 759 ms (Sample #2) and 4 839 ms (Sample #1).

Table 5.13: Time to classify records on single server and on multiple servers (Sample #1)

	Single Server	**Multiple Servers**		
Model	**Server_0**	**Server_1**	**Server_2**	**Server_3**
DTC	998 ms	995 ms		
RFC	4 863 ms		4 839 ms	
SLog	1 149 ms			1 152 ms
Total	7 010 ms	4 839 ms		

Table 5.14: Time to classify records on single server and on multiple servers (Sample #2)

	Single Server	**Multiple Servers**		
Model	**Server_0**	**Server_1**	**Server_2**	**Server_3**
DTC	991 ms	999 ms		
RFC	4 724 ms		4 759 ms	
SLog	1 348 ms			1 301 ms
Total	7 063 ms	4 759 ms		

Table 5.15: Time to classify records on single server and on multiple servers (Sample #3)

	Single Server	**Multiple Servers**		
Model	**Server_0**	**Server_1**	**Server_2**	**Server_3**
DTC	1 032 ms	1 001 ms		
RFC	4 835 ms		4 812 ms	
SLog	1 201 ms			1 160 ms
Total	7 068 ms	4 812 ms		

Table 5.16: Time to classify records on single server and on multiple servers (Sample #4)

	Single Server	**Multiple Servers**		
Model	**Server_0**	**Server_1**	**Server_2**	**Server_3**
DTC	1 004 ms	1 000 ms		
RFC	4 801 ms		4 787 ms	
SLog	1 253 ms			1 201 ms
Total	7 058 ms	4 787 ms		

Table 5.17 presents the variation of the time used for classification in each Sample in a single server and in a multiple servers. Fig. 5.4 draws the results of Table 5.17. From these results, it can be verified that when multiple servers are used, there is a time reduction between 31% and 33% compared to a single server.

Table 5.17: Variation of the time used for classification

	Single Server (S.S.) (ms)	**Multiple Servers (M.S.) (ms)**	**M.S. - S.S. (%)**
Sample #1	7 010	4 839	- 30.97
Sample #2	7 063	4 759	- 32.62
Sample #3	7 068	4 812	- 31.92
Sample #4	7 058	4 787	- 32.18

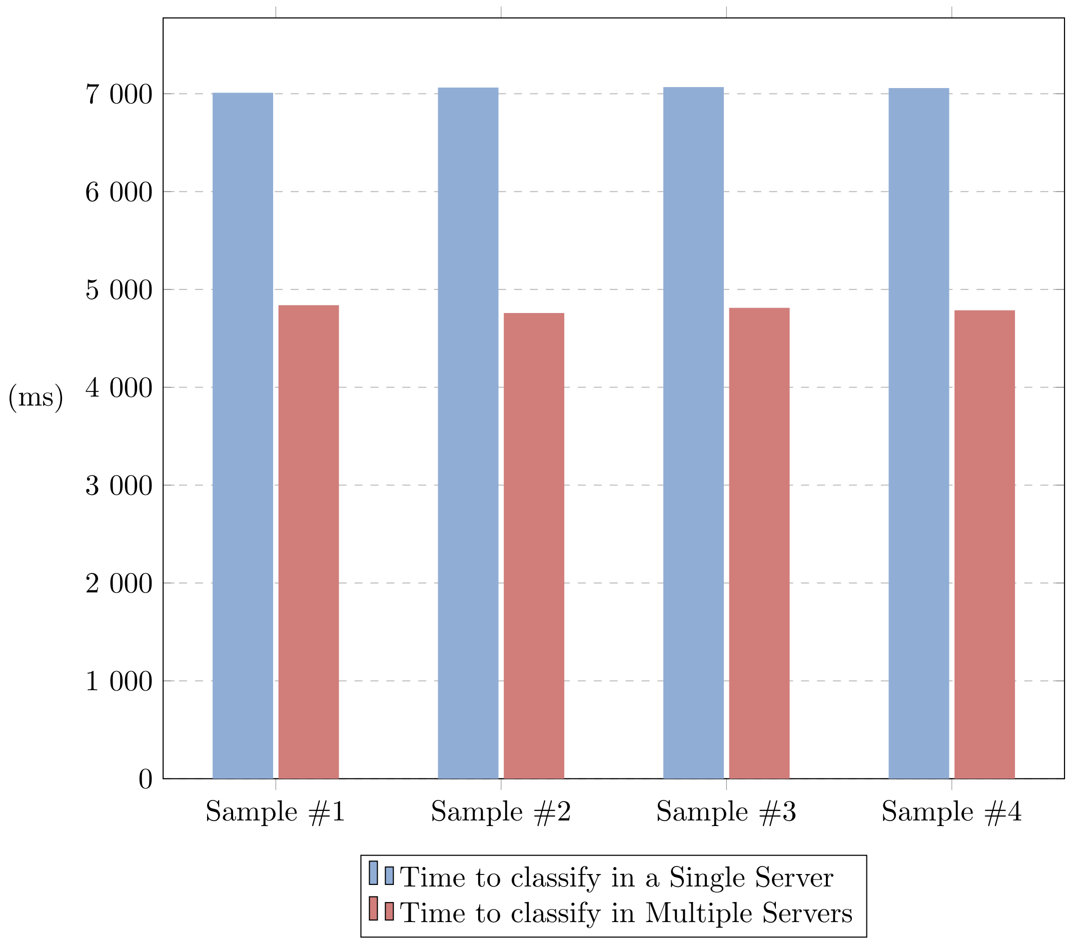


Figure 5.4: Results of time to classify subsets in one server and in multiple servers

Table 5.18, Table 5.19, Table 5.20, and Table 5.21 presents the time taken to training a subset with 200k records on a single server (without ray) and on multiple servers (distributed tasks managed by ray), in four samples or iterations, i.e. Sample #1, Sample #2, Sample #3, Sample #4. From the results obtained, it can be verified that as in a single server (Server_0) the training in the different models are carried out in sequence, the total time represents the sum of each training, so between 105 814 ms (Sample #1) and 106 228 ms (Sample #2). In turn, when using multiple servers (Server_1, Server_2, and Server_3), i.e. with task distribution, the training runs in parallel, and the total time is equal to the longest training, thus between 60 103 ms (Sample #2) and 60 999 ms (Sample #3).

Table 5.18: Time to train subsets on single server and on multiple servers (Sample #1)

	Single Server	**Multiple Servers**		
Algorithm	**Server_0**	**Server_1**	**Server_2**	**Server_3**
DTC	2 004 ms	1 998 ms		
RFC	60 683 ms		60 591 ms	
SLog	43 127 ms			43 419 ms
Total	105 814 ms	60 591 ms		

Table 5.19: Time to train subsets on single server and on multiple servers (Sample #2)

	Single Server	**Multiple Servers**		
Algorithm	**Server_0**	**Server_1**	**Server_2**	**Server_3**
DTC	2 418 ms	2 193 ms		
RFC	60 391 ms		60 103 ms	
SLog	43 419 ms			43 399 ms
Total	106 228 ms	60 103 ms		

Table 5.20: Time to train subsets on single server and on multiple servers (Sample #3)

	Single Server	**Multiple Servers**		
Algorithm	**Server_0**	**Server_1**	**Server_2**	**Server_3**
DTC	1 991 ms	1 999 ms		
RFC	61 028 ms		60 999 ms	
SLog	43 098 ms			42 897 ms
Total	106 117 ms	60 999 ms		

Table 5.21: Time to train subsets on single server and on multiple servers (Sample #4)

	Single Server	**Multiple Servers**		
Algorithm	**Server_0**	**Server_1**	**Server_2**	**Server_3**
DTC	2 193 ms	2 154 ms		
RFC	60 571 ms		60 721 ms	
SLog	43 352 ms			43 004 ms
Total	106 116 ms	60 721 ms		

Table 5.22 presents the variation of the time used for training in each sample in a single server and in a multiple servers. Fig. 5.5 draws the results of Table 5.22. From these results, it can be verified that when multiple servers are used, there is a time reduction of 43% compared to a single server.

Table 5.22: Variation of the time used for training

	Single Server (S.S.) (ms)	**Multiple Servers (M.S.) (ms)**	**M.S. - S.S. (%)**
Sample #1	105 814	60 591	- 42.74
Sample #2	106 228	60 103	- 43.42
Sample #3	106 117	60 999	- 42.49
Sample #4	106 116	60 721	- 42.78

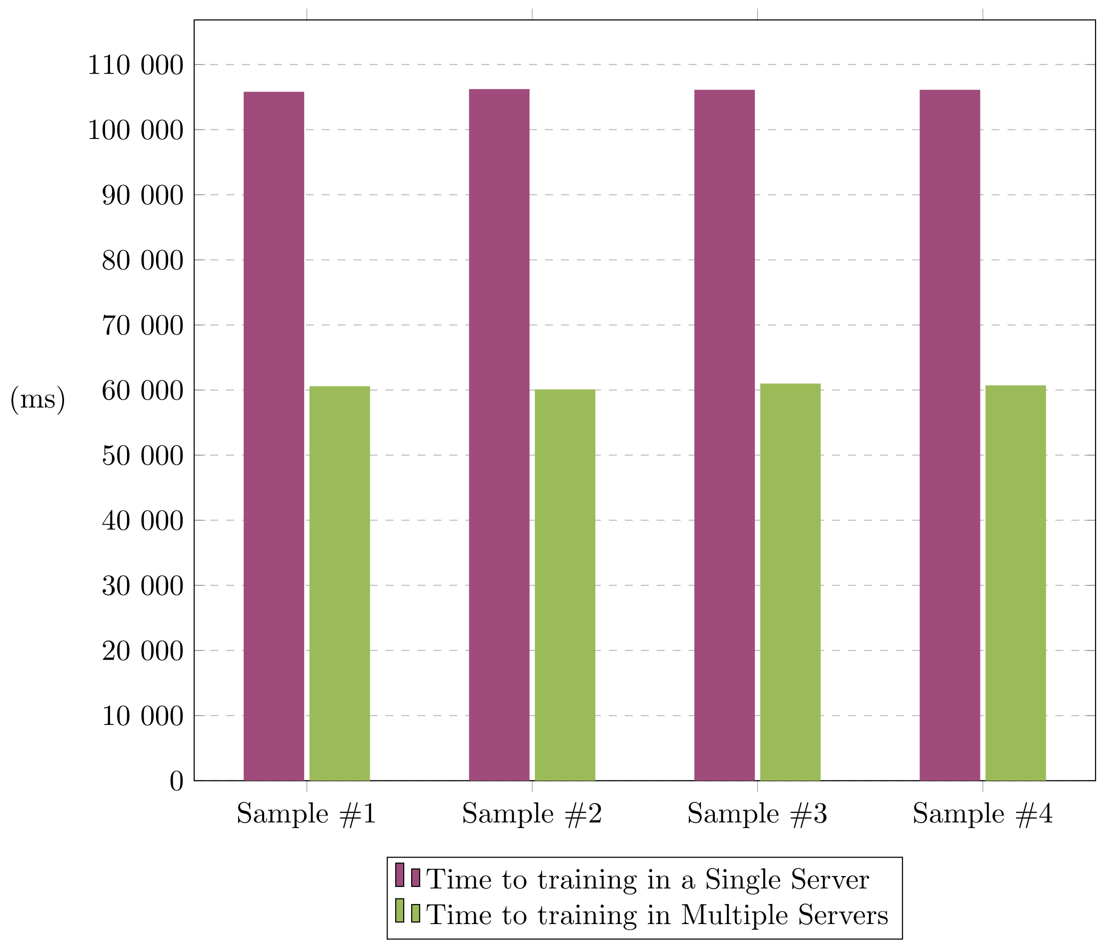


Figure 5.5: Results of time to training subsets in one server and in multiple servers

From the results presented in Table 5.17, Fig. 5.4, Table 5.22, and Fig. 5.5 it can be verified the relevance of implementing this architecture in Fed4FIRE+ with the execution of tasks in multiple servers managed by Ray. In the classification stages, a reduction in time of between 31% and 33% was obtained. In the training stages, a reduction of approximately 43% was obtained.

Chapter 6

Conclusions

An IDS assists systems administrators by preventing and actuating on potential threats to systems and data. Anomaly-based IDS can use machine learning algorithms to classify events either as normal or anomalous. When using Supervised Learning, these algorithms learn how to classify records from classified datasets. In order to improve the performance of the classification algorithms, the datasets should be recent, contain data from different sources in a collaborative approach, i.e. from different companies, and trained with multiple algorithms.

In this document, a centralized and vote-based architecture is proposed to generate classified datasets and improve performance of a supervised learning-based IDS. The proposed architecture uses records from multiple IDS that are classified with multiple models, and it uses a majority vote system to generate richer and classified datasets. These datasets are then used to train and the best models by service are deployed in each IDS.

The performance of five machine learning algorithms (Decision Tree Classifier, Random Forest Classifier, K-Nearest Neighbors, Simple Logistics, and Support Vector Machine) was assessed regarding their accuracy, precision, and elapsed runtime values, and three algorithms (Decision Tree Classifier, Random Forest Classifier, and Simple Logistics) were selected to validate the architecture.

Five samples of the CICDDoS2019 dataset were prepared and used in a testbed designed and deployed to assess the proposed architecture against a baseline scenario. From the results obtained, the proposed architecture was able to generate classified datasets and to choose the best model in each iteration, enabling an increase of 11.5% in accuracy value and an increase of 11.2% in the precision value in the four tested iterations, when compared to the baseline scenario.

The proposed architecture assumes recurrent classification and training with multiple algorithms tasks, which extends the overall execution times and requires resources. In order to expediently run these tasks and apply the best model in each company, the classification and training tasks of the proposed architecture were implemented in Fed4FIRE+, with Ray to distribute tasks by the available resources. The use of Fed4FIRE+ resources and Ray allowed a reduction of the classification time between 31% and 33%, and of the training time by around 43%.

Future efforts may be developed to test the proposed architecture with other machine learning algorithms and against other types of attacks and datasets. Furthermore, internal procedures could be changed to dynamically balance the trade-off between complexity and elapsed execution time against detection performance.

References

[1] "The world's most valuable resource is no longer oil, but data". In: *Economist* (May 2017).

[2] George Grispos. "Criminals: Cybercriminals". In: *Encyclopedia of Security and Emergency Management* November (2019). DOI: 10.1007/978-3-319-69891-5.

[3] Accenture. *State of Cybersecurity Resilience 2021*. 2021. URL: https://www.accenture.com/_acnmedia/PDF-165/Accenture-State-Of-Cybersecurity-2021.pdf.

[4] Thanh Cong Truong, Quoc Bao Diep, and Ivan Zelinka. "Artificial Intelligence in the Cyber Domain: Offense and Defense". In: *Symmetry* 12.3 (2020). ISSN: 2073-8994. DOI: 10.3390/sym12030410. URL: https://www.mdpi.com/2073-8994/12/3/410.

[5] Manik Deep Singh. "Analysis of Host-Based and Network-Based Intrusion Detection System". In: *Computer Network and Information Security* 8 (2014), pp. 41–47. DOI: 10.5815/ijcnis.2014.08.06. URL: https://doi.org/10.5815/ijcnis.2014.08.06.

[6] VVRPV Jyothsna, Rama Prasad, and K Munivara Prasad. "A review of anomaly based intrusion detection systems". In: *International Journal of Computer Applications* 28.7 (2011), pp. 26–35.

[7] Rohit Kumar Singh Gautam and Er Amit Doegar. "An Ensemble Approach for Intrusion Detection System Using Machine Learning Algorithms". In: *Proceedings of the 8th International Conference Confluence 2018 on Cloud Computing, Data Science and Engineering, Confluence 2018* (2018), pp. 61–64. DOI: 10.1109/CONFLUENCE.2018.8442693.

[8] Sreenivas Sremath Tirumala, Hira Sathu, and Abdolhossein Sarrafzadeh. “Free and open source intrusion detection systems: A study”. In: *2015 International Conference on Machine Learning and Cybernetics (ICMLC)*. Vol. 1. 2015, pp. 205–210. DOI: 10.1109/ICMLC.2015.7340923.

[9] Gulshan Kumar and Krishan Kumar. “The Use of Artificial-Intelligence-Based Ensembles for Intrusion Detection: A Review”. In: *Applied Computational Intelligence and Soft Computing* (2012). DOI: 10.1155/2012/850160.

[10] Safwan Mawlood Hussein. “Performance Evaluation of Intrusion Detection System Using Anomaly and Signature Based Algorithms to Reduction False Alarm Rate and Detect Unknown Attacks”. In: *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2016, pp. 1064–1069. DOI: 10.1109/CSCI.2016.0203.

[11] Tameem Ahmad, Mohd Asad Anwar, and Misbahul Haque. “Machine Learning Techniques for Intrusion Detection”. In: (2013), pp. 47–65. DOI: 10.4018/978-1-7998-2242-4.ch003. arXiv: 1312.2177.

[12] Babak Khosravifar and Jamal Bentahar. “An Experience Improving Intrusion Detection Systems False Alarm Ratio by Using Honeypot”. In: *22nd International Conference on Advanced Information Networking and Applications (aina 2008)*. 2008, pp. 997–1004. DOI: 10.1109/AINA.2008.44.

[13] R. Venkatesan et al. “A Novel Approach for Detecting DDoS Attack in H-IDS Using Association Rule”. In: *2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA)*. 2018, pp. 1–5. DOI: 10.1109/ICSCAN.2018.8541174.

[14] Diogo Teixeira et al. “OSSEC IDS Extension to Improve Log Analysis and Override False Positive or Negative Detections”. In: *Journal of Sensor and Actuator Networks 2019, Vol. 8, Page 46* 8.3 (Sept. 2019), p. 46. DOI: 10.3390/JSAN8030046. URL: https://www.mdpi.com/2224-2708/8/3/46/htm.

[15] Xianwei Gao et al. “An Adaptive Ensemble Machine Learning Model for Intrusion Detection”. In: *IEEE Access* 7 (2019), pp. 82512–82521. DOI: 10.1109/ACCESS.2019.2923640.

[16] Mrutyunjaya Panda and Manas Patra. "Ensemble voting system for anomaly based network intrusion detection". In: *FULL PAPER International Journal of Recent Trends in Engineering* 2 (Jan. 2009).

[17] Vikas C. Raykar et al. "Supervised Learning from Multiple Experts: Whom to Trust When Everyone Lies a Bit". In: *Proceedings of the 26th Annual International Conference on Machine Learning.* Association for Computing Machinery, 2009, pp. 889–896. ISBN: 9781605585161. DOI: 10.1145/1553374.1553488.

[18] Mario Di Mauro and Cesario Di Sarno. "Improving SIEM capabilities through an enhanced probe for encrypted Skype traffic detection". In: *J. Inf. Secur. Appl.* 38 (2018), pp. 85–95.

[19] Ahmed Mahfouz et al. "Ensemble Classifiers for Network Intrusion Detection Using a Novel Network Attack Dataset". In: *Future Internet* 12.11 (2020). ISSN: 1999-5903. DOI: 10.3390/fi12110180. URL: https://www.mdpi.com/1999-5903/12/11/180.

[20] Mohammad Hashem Haghighat and Jun Li. "Intrusion detection system using voting-based neural network". In: *Tsinghua Science and Technology* 26.4 (2021), pp. 484–495. DOI: 10.26599/TST.2020.9010022.

[21] Kishor Kumar Gulla et al. "Machine learning based intrusion detection techniques". In: *Handbook of Computer Networks and Cyber Security: Principles and Paradigms* Icoei (2019), pp. 873–888. DOI: 10.1007/978-3-030-22277-2_35.

[22] L. Haripriya and M. A. Jabbar. "Role of Machine Learning in Intrusion Detection System: Review". In: *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018* Iceca (2018), pp. 925–929. DOI: 10.1109/ICECA.2018.8474576.

[23] Iksoo Shin et al. "Platform design and implementation for flexible data processing and building ML models of IDS alerts". In: *Proceedings - 2019 14th Asia Joint Conference on Information Security, AsiaJCIS 2019* (2019), pp. 64–71. DOI: 10.1109/AsiaJCIS.2019.000-4.

[24] Aditya Vikram and Mohana. "Anomaly detection in Network Traffic Using Unsupervised Machine learning Approach". In: *2020 5th International Conference on Com-*

munication and Electronics Systems (ICCES). 2020, pp. 476–479. DOI: 10.1109/ICCES48766.2020.9137987.

[25] Eirini Anthi et al. “A Supervised Intrusion Detection System for Smart Home IoT Devices”. In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 9042–9053. DOI: 10.1109/JIOT.2019.2926365.

[26] Kazi Abu Taher, Billal Mohammed Yasin Jisan, and Md. Mahbubur Rahman. “Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection”. In: *2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)*. 2019, pp. 643–646. DOI: 10.1109/ICREST.2019.8644161.

[27] Aamir S Ahanger, Sajad M Khan, and Faheem Masoodi. “An Effective Intrusion Detection System using Supervised Machine Learning Techniques”. In: *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*. 2021, pp. 1639–1644. DOI: 10.1109/ICCMC51019.2021.9418291.

[28] Deepa Rani and Narottam Chand Kaushal. “Supervised Machine Learning Based Network Intrusion Detection System for Internet of Things”. In: *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. 2020, pp. 1–7. DOI: 10.1109/ICCCNT49239.2020.9225340.

[29] Guochen Shi and Gang He. “Collaborative Multi-agent Reinforcement Learning for Intrusion Detection”. In: *2021 7th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC)*. 2021, pp. 245–249. DOI: 10.1109/IC-NIDC54101.2021.9660402.

[30] Shahid Latif et al. “Intrusion Detection Framework for the Internet of Things using a Dense Random Neural Network”. In: *IEEE Transactions on Industrial Informatics* (2021). DOI: 10.1109/TII.2021.3130248.

[31] Kunal and Mohit Dua. “Machine Learning Approach to IDS: A Comprehensive Review”. In: *Proceedings of the 3rd International Conference on Electronics and Communication and Aerospace Technology, ICECA 2019* (2019), pp. 117–121. DOI: 10.1109/ICECA.2019.8822120.

[32] Weiyang Mo et al. “Deep-neural-network-based wavelength selection and switching in ROADM systems”. In: *Journal of Optical Communications and Networking* 10 (2018). ISSN: 19430620. DOI: 10.1364/JOCN.10.0000D1.

[33] Silvestre Malta., Pedro Pinto., and Manuel Veiga. “Using Syntactic Similarity to Shorten the Training Time of Deep Learning Models using Time Series Datasets: A Case Study”. In: *Proceedings of the 2nd International Conference on Deep Learning Theory and Applications - DeLTA,* INSTICC. SciTePress, 2021, pp. 93–100. ISBN: 978-989-758-526-5. DOI: 10.5220/0010515700930100.

[34] Joost Verbraeken et al. “A Survey on Distributed Machine Learning”. In: *ACM Computing Surveys* 53.2 (2020), pp. 1–33. ISSN: 15577341. DOI: 10.1145/3377454. URL: http://arxiv.org/abs/1912.09789.

[35] Fed4FIRE+. *About Fed4FIRE+.* 2022. URL: https://www.fed4fire.eu/the-project/ (visited on 01/10/2022).

[36] Global Environment for Network Innovations (GENI). *What is GENI?* 2022. URL: https://www.geni.net/about-geni/what-is-geni/ (visited on 01/10/2022).

[37] *Smart Applications on Virtual Infrastructure (SAVI).* 2022. URL: https://www.savinetwork.ca/ (visited on 01/10/2022).

[38] TaeHong Kim et al. “Survey and Performance Test of Python-Based Libraries for Parallel Processing”. In: *The 9th International Conference on Smart Media and Applications.* SMA 2020. Association for Computing Machinery, 2020, pp. 154–157. ISBN: 9781450389259. DOI: 10.1145/3426020.3426057. URL: https://doi.org/10.1145/3426020.3426057.

[39] Ray. *What is Ray?* 2021. URL: https://docs.ray.io/en/master/ (visited on 01/12/2022).

[40] *Using IPython for parallel computing.* URL: https://ipyparallel.readthedocs.io/en/latest/ (visited on 01/12/2022).

[41] *dispy: Distributed and Parallel Computing with/for Python — dispy 4.12.0 documentation.* URL: http://dispy.sourceforge.net/ (visited on 01/12/2022).

[42] *Pandaral·lel.* URL: https://github.com/nalepae/pandarallel/tree/v1.5.4 (visited on 01/12/2022).

[43] Dask. *Dask - Documentation.* 2022. URL: https://docs.dask.org/en/latest/ (visited on 01/12/2022).

[44] Joblib. *Joblib: running Python functions as pipeline jobs.* 2022. URL: https://joblib.readthedocs.io/en/latest/ (visited on 01/12/2022).

[45] Philipp Moritz et al. "Ray: A Distributed Framework for Emerging AI Applications". In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18).* USENIX Association, Oct. 2018, pp. 561–577. ISBN: 978-1-939133-08-3. URL: https://www.usenix.org/conference/osdi18/presentation/moritz.

[46] Enzo Fabbiani et al. "Distributed Big Data Analysis for Mobility Estimation in Intelligent Transportation Systems". In: *High Performance Computing.* 2016, pp. 146–160. ISBN: 978-3-319-57972-6. DOI: https://doi.org/10.1007/978-3-319-57972-6_11.

[47] *KDD Cup 1999.* 1999. URL: http://kdd.ics.uci.edu/databases/kddcup (visited on 12/21/2021).

[48] *Nsl-kdd dataset.* 2014. URL: https://www.unb.ca/cic/datasets/nsl.html (visited on 12/21/2021).

[49] *The UNSW-NB15 Dataset Description.* 2015. URL: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/ (visited on 12/21/2021).

[50] Iman Sharafaldin et al. "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy". In: *Proceedings - International Carnahan Conference on Security Technology* 2019-October.Cic (2019). ISSN: 10716572. DOI: 10.1109/CCST.2019.8888419.

[51] kaspersky. *What is a DDoS Attack? - DDoS Meaning.* 2022. URL: https://www.kaspersky.com/resource-center/threats/ddos-attacks.

[52] SecurityWeek Eduard Kovacs. *RPC Portmapper Abused for DDoS Attack Reflection, Amplification.* 2015. URL: https://www.securityweek.com/rpc-portmapper-abused-ddos-attack-reflection-amplification.

[53] Kishore Babu Dasari and Nagaraju Devarakonda. "Detection of Different DDoS Attacks Using Machine Learning Classification Algorithms". In: *Ingénierie des systèmes d information* 26.5 (Oct. 2021), pp. 461–468. DOI: 10.18280/isi.260505. URL: https://doi.org/10.18280/isi.260505.

[54] Sanchita Pradhan. "Internet and Society: Latest Developments in Cyberspace". In: *Nirma University Law Journal* 6.2 (Dec. 2018). URL: https://ssrn.com/abstract=3442953.

[55] Rizgar R. Zebari, Subhi R. M. Zeebaree, and Karwan Jacksi. "Impact Analysis of HTTP and SYN Flood DDoS Attacks on Apache 2 and IIS 10.0 Web Servers". In: *2018 International Conference on Advanced Science and Engineering (ICOASE).* 2018, pp. 156–161. DOI: 10.1109/ICOASE.2018.8548783.

[56] Saif ur Rehman et al. "DIDDOS: An approach for detection and identification of Distributed Denial of Service (DDoS) cyberattacks using Gated Recurrent Units (GRU)". In: *Future Generation Computer Systems* 118 (2021), pp. 453–466. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2021.01.022. URL: https://www.sciencedirect.com/science/article/pii/S0167739X21000327.

[57] Arvind Prasad and Shalini Chandra. "VMFCVD: An Optimized Framework to Combat Volumetric DDoS Attacks using Machine Learning". In: *Arabian Journal for Science and Engineering* (2022). DOI: 10.1007/s13369-021-06484-9.

[58] Nisar Memon, Intesab Hussain, and Zahid Yousaf. "Analysis and Detection of DDoS Attacks Targetting Virtualized Servers". In: (July 2019).

[59] Celestine Iwendi et al. "Sustainable Security for the Internet of Things Using Artificial Intelligence Architectures". In: *ACM Trans. Internet Technol.* 21.3 (June 2021). ISSN: 1533-5399. DOI: 10.1145/3448614.

[60] Rami J. Alzahrani and Ahmed Alzahrani. "Security Analysis of DDoS Attacks Using Machine Learning Algorithms in Networks Traffic". In: *Electronics* 10.23 (2021).

ISSN: 2079-9292. DOI: 10.3390/electronics10232919. URL: https://www.mdpi.com/2079-9292/10/23/2919.

[61] Mateo Florez Cardenas and Gabriel Acar. *Ethical Hacking of a Smart Fridge : Evaluating the cybersecurity of an IoT device through gray box hacking.* 2021.

[62] Matheus P. Novaes et al. “Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments”. In: *Future Generation Computer Systems* 125 (2021). ISSN: 0167-739X. DOI: 10.1016/j.future.2021.06.047. URL: https://www.sciencedirect.com/science/article/pii/S0167739X21002429.

[63] Junhong Li. “Detection of DDoS Attacks Based On Dense Neural Networks, Autoenconders And Pearson Correlation Coefficient”. In: (2020). URL: http://hdl.handle.net/10222/78536.