*future internet*

MDPI

*Article*

# A Vote-Based Architecture to Generate Classified Datasets and Improve Performance of Intrusion Detection Systems Based on Supervised Learning

**Diogo Teixeira [1,*], Silvestre Malta [1] and Pedro Pinto [1,2,3]**

[1] Instituto Politécnico de Viana do Castelo, 4900-347 Viana do Castelo, Portugal; smalta@estg.ipvc.pt (S.M.); pedropinto@estg.ipvc.pt (P.P.)
[2] Instituto Universitário da Maia, 4475-690 Maia, Portugal
[3] INESC TEC—Institute for Systems and Computer Engineering, Technology and Science, 4200-465 Porto, Portugal
* Correspondence: diogoteixeira@ipvc.pt

**Abstract:** An intrusion detection system (IDS) is an important tool to prevent potential threats to systems and data. 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 paper proposes a vote-based architecture to generate classified datasets and improve the performance of supervised learning-based IDSs. On a regular basis, multiple IDSs in different locations 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. The proposed architecture trains multiple times with several algorithms. To shorten the overall runtimes, the proposed architecture was deployed in Fed4FIRE+ with Ray to distribute the tasks by the available resources. 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; accuracy; precision; IDS



**Citation:** Teixeira, D.; Malta, S.; Pinto, P. A Vote-Based Architecture to Generate Classified Datasets and Improve Performance of Intrusion Detection Systems Based on Supervised Learning. *Future Internet* **2022**, *14*, 72. https://doi.org/10.3390/fi14030072

Academic Editor: Izzat Alsmadi

Received: 2 February 2022
Accepted: 22 February 2022
Published: 25 February 2022



## 1. Introduction

Cyberattacks are constantly performed against companies and institutions, and these criminal activities may have different objectives, such as to disrupt services, steal confidential information, or perform extortion [1]. The complexity and intelligence of attacks evolves and, therefore, defense systems need to keep the pace to be effective [2]. 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), and intrusion prevention system (IPS), among others.

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 [3]. IDSs monitor a network or system, and their detection method can be classified as a 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 defined pattern [4]. These patterns and anomalies can be tested using machine learning algorithms.

Machine learning algorithms are used to train models using a given environment or dataset. The learning process of machine learning algorithms can be divided into three groups: unsupervised learning, supervised learning and reinforcement learning. In supervised learning, the algorithm learns how to classify records from a labeled or classified dataset. In unsupervised learning, the algorithm learns or draws inferences from an unlabeled or unclassified dataset. In reinforcement learning, the algorithm interacts with the environment to make decisions, and receives a reward for correct decisions and a penalty for errors.

A company wishing to protect its networks, devices or services with an anomaly-based IDS may use a supervised learning algorithm. However, for the 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 is trained with multiple algorithms.

In this paper is proposed a centralized and vote-based architecture to generate classified datasets and improve the performance of supervised learning-based intrusion detection systems. The proposed architecture is presented in Figure 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 obtain 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 the IDS performance.

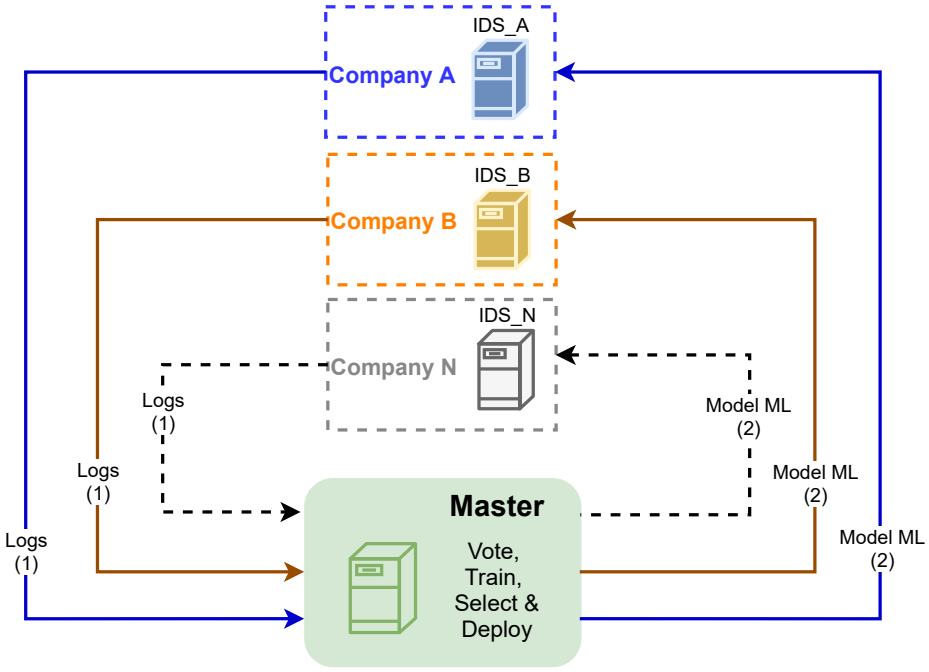


**Figure 1.** Proposed architecture.

The architecture is ready to use multiple algorithms to classify new records, and its voting system ensures that these records are classified according to a majority decision, thus increasing the reliability of the classification process. The generation of a dataset using recent and diverse records enables to enrich the dataset, improving the accuracy and precision of IDS over time. This architecture assumes the intensive and scalable training of models, and this takes time and resources. To cope with these requirements and distribute the training tasks, the proposed architecture is deployed with Fed4FIRE+ [5], a federation of testbeds with scalable processing resources.

This paper is organized as follows. In Section 2, the related work is presented. Section 3 details the proposed architecture and its operation. Section 4 presents the validation tests, the results and the analysis of the results. In Section 5, the conclusions are presented.

## 2. Related Work

An IDS protects corporations and institutions against cyberattacks by monitoring multiple events in different agents and triggering alerts or actions [6]. IDSs can be classified into two different types: 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. The NIDS operates in a network where packets are captured and analyzed to enable the implementation of adequate protective measures. Ref. [7] presented a detailed study of these two types. It was concluded that the best choice is dependent on purpose, risk and features. Regarding their detection method, IDS can be categorized as signature based or anomaly based. In [8], a performance evaluation of these modes is presented. The conclusion presented shows that both methods have a limitation to detect known and unknown attacks.

As cyberattacks are constantly evolving, so IDS must also progress to be efficient [9]. A set of research works are focused on improving the operation and performance of the IDS. Authors in [10] proposed an architecture to reduce the false alarm rate of the attack detection. Authors in [11] proposed to prevent distributed denial of service (DDoS) attacks by using the configuration features and rule adjustments of OSSEC and described the operation of an algorithm used to distinguish real from false DDoS alerts. In order to protect a set of machines, Teixeira et al. [12] 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 [13] proposed 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. [14] designed a voting system to detect errors and concluded that it performs efficiently in terms of a high detection rate and low false positive rate. Authors in [15,16] implemented a voting system that uses probability mechanisms to define the final classification. Mahfouz et al. [17] proposed an ensemble classifier model that includes a voting system to improve the detection accuracy and true positive rate and decrease the false positive rate. Authors in [18] proposed 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 [19]. According to Haripriya et al. [20], 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 [21], using machine learning techniques in an IDS can reduce the occurrence of false positives. The authors also pointed out that one or more models should be used to increase the performance of their detection. In [22], Vikram et al. implemented 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 [23], Anthi et al. proposed 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 [24], the 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 [25], four algorithms are compared to implement on IDS, and all of them have accuracy greater than 96%. In [26], Rani et al. proposed an efficient method with a uniform detection system based on the supervised machine learning technique that obtained an accuracy of 99.9%. In [27], the authors implemented collaborative multi-agent reinforcement learning to make the detection more efficient. The results are better in comparison with the baseline approach. In [28], 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 [29], 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 IDSs to generate new datasets and improve their performance in a collaborative approach.

The implementation of machine learning algorithms requires time and resources to train. As highlighted by Mo et al. [30], a long training time can add significant costs. Long training running time of the models leads to the use of distributed systems for an increase in parallelization and total amount of I/O bandwidth [31]. Federated testbed platforms exist for implementation, validation, and testing. The most considerable are Fed4FIRE+ [5], GENI [32] and SAVI [33]. 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 [34] surveyed and tested the following open-source Python-based libraries for parallel processing: Ray [35], Ipyparallel [36], Dispy [37], Pandaral-lel [38], Dask [39], and Joblib [40]. Authors in [41] 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 [42], Dispy was used to manage distributing parallel tasks among several computer nodes to decrease the execution time of specific tasks.

## 3. The Proposed Architecture

In Figure 2, the general operation of the proposed architecture is presented. In a set of companies there is an IDS collecting unlabeled service logs from multiple agents of different service types (e.g., FTP, MySQL, and 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 using "attack" or "ok") using different machine learning models. This 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).

The operation of the master assumes a set of classifiers and trainers, whose number can be adjusted according to 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 Figure 3. An external and categorized dataset 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).

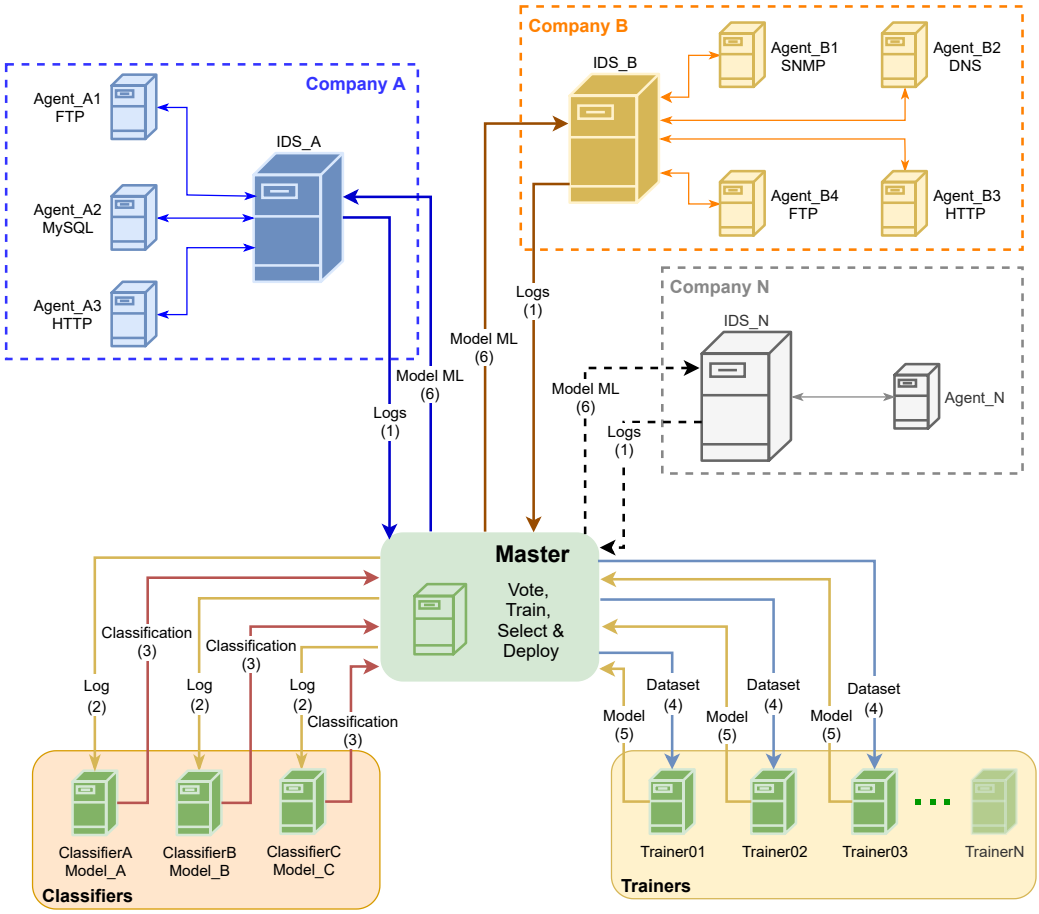


**Figure 2.** General architecture.

After the first iteration, the proposed architecture is ready to operate on a regular basis. The companies start by collecting the service logs and sending them to the master. Figure 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, and MySQL). These logs are sent to the IDS of the company, which is composed of multiple anomaly-based IDSs, using machine learning, with a model for each service type. These models classify each service log 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.

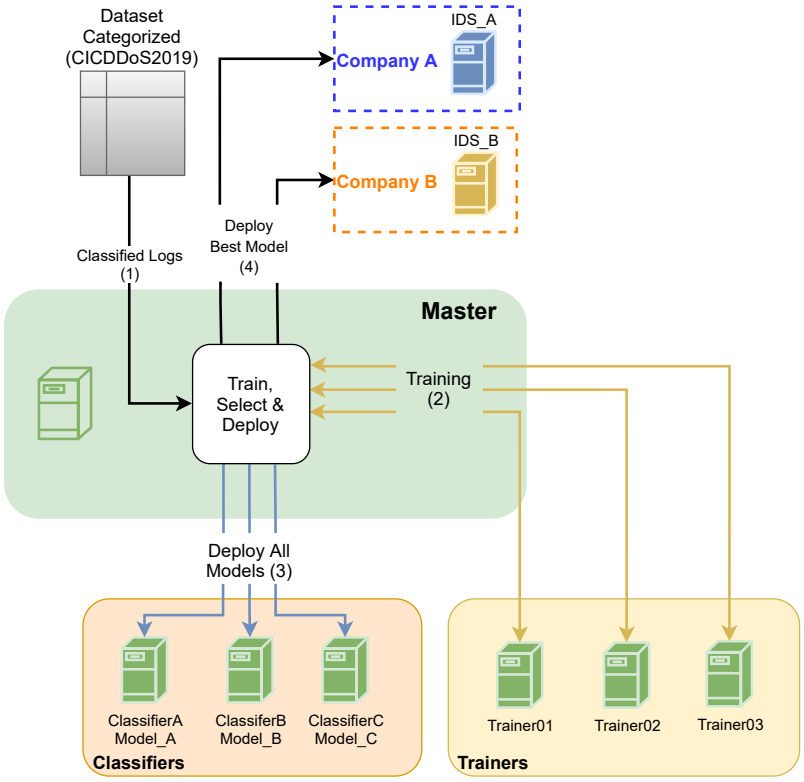


**Figure 3.** Initial operation.

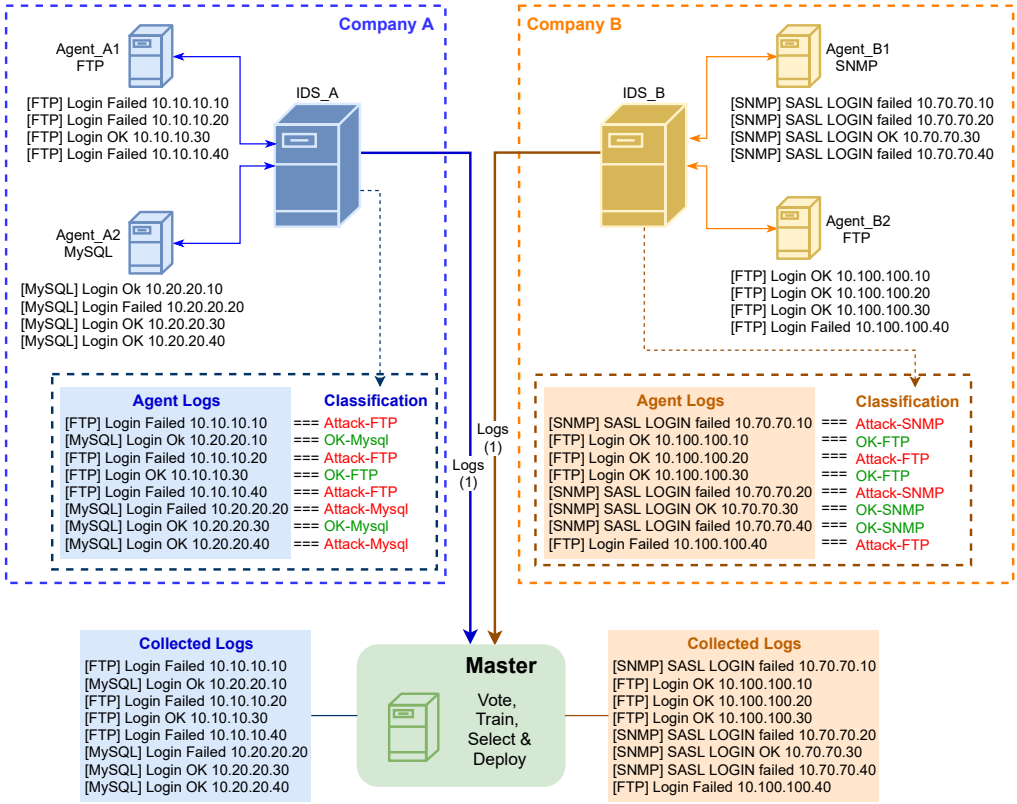


**Figure 4.** Unclassified logs collection from companies' IDSs.

These unclassified logs are then received by the master, and its internal procedures are depicted in Figure 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., Classifiers A, B and so on). Each classifier has 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 100,000 classified records), the training orchestrator sends the generated dataset to a set of trainers (5) (i.e., Trainers 1, 2, and so on). 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).

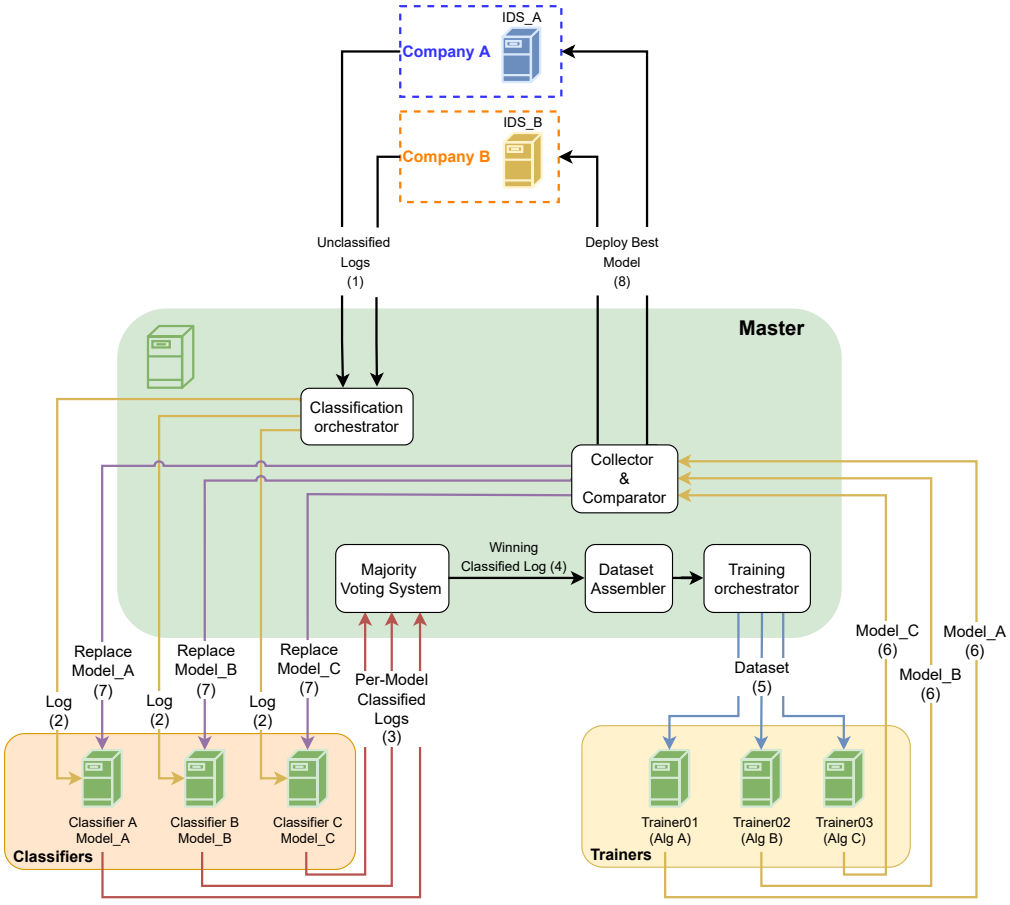


**Figure 5.** Master internal procedures.

The master's internal procedures for classification, voting and dataset generation are depicted in Figure 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); it replaces models in the classifiers (7) and deploys the best models (8) in the IDS of the companies.

In particular, the training, selection and deployment stages comprehend three phases, depicted in Figure 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 are used in the voting system that classifies the most recent records received from the companies as shown previously in Figure 6. Finally, in phase 3, the model with better accuracy and precision is sent to the IDSs of companies to increase their performance in detecting attacks.

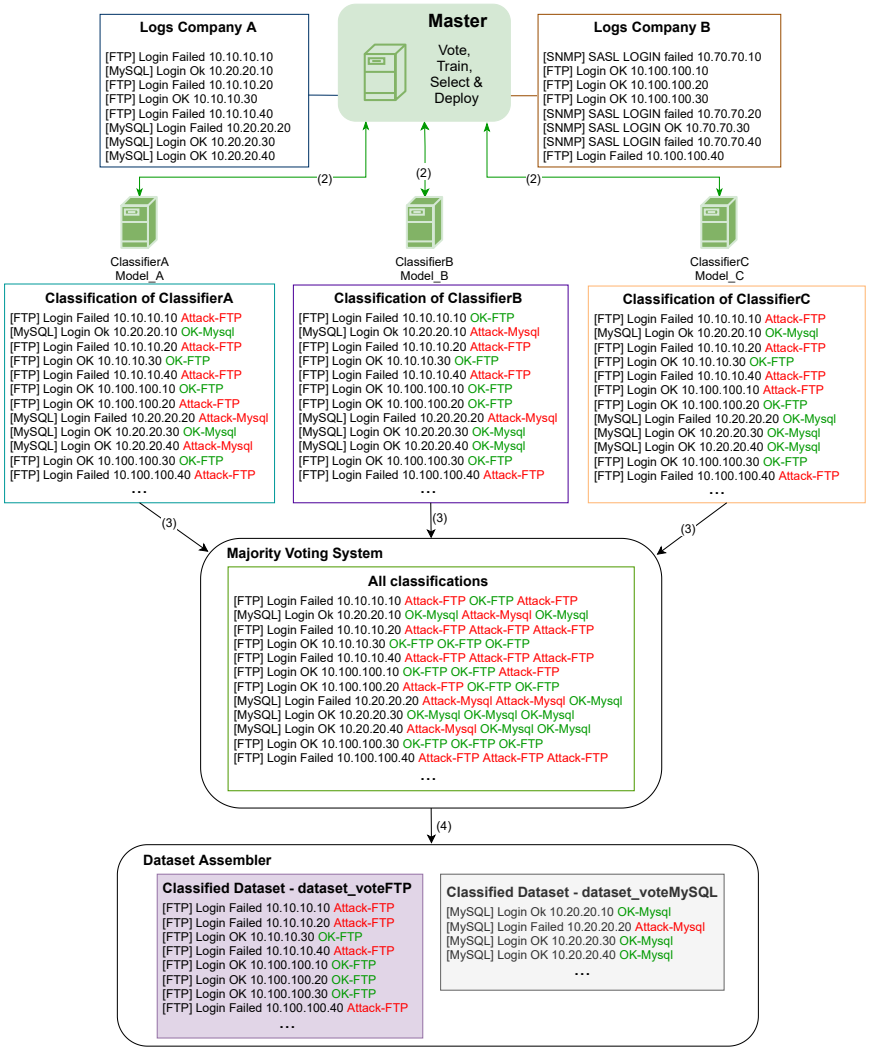


**Figure 6.** Master internal procedures for classification, voting and dataset generation.

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.

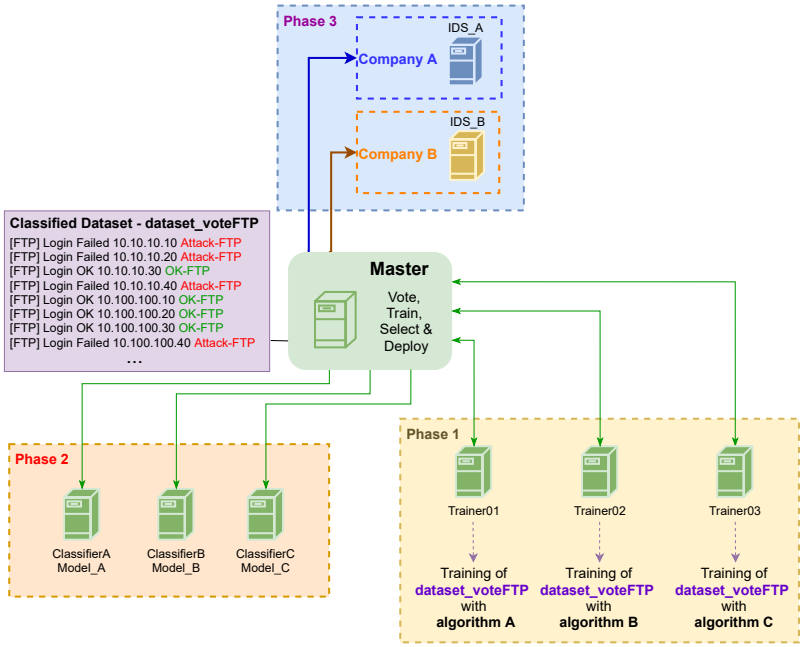


**Figure 7.** New training, selection and deployment.

## 4. Results and Analysis

The proposed architecture was evaluated in a set of validation tests. The initial operation requires a classified dataset and thus, datasets such as NSL-KDD [43] (based on KDD'99 [44]), UNSW-NB15 [45], or CICDDoS2019 [46] can be used. These datasets were built with records from 2009, 2015, and 2019, respectively. In the validation tests, the most recent one, CICDDoS2019, was used. This dataset features 50 million records from 2019, distributed by 14 different labels, according to Table 1.

**Table 1.** Number of instances in the CICDDoS2019 dataset.

| Attribute (Class Label) | Number of Instances |
|---|---|
| DDoS_WebDDoS | 439 |
| Benign (legitimate traffic) | 56,863 |
| DDoS_Portmap | 186,960 |
| DDoS_UDP-Lag | 366,461 |
| DDoS_NTP | 1,202,642 |
| DDoS_SYN | 1,582,289 |
| DDoS_LDAP | 2,179,930 |
| DDoS_SSDP | 2,610,611 |
| DDoS_UDP | 3,134,645 |
| DDoS_NetBIOS | 4,093,279 |
| DDoS_MSSQL | 4,522,492 |
| DDoS_DNS | 5,071,011 |
| DDoS_SNMP | 5,159,870 |
| DDoS_TFTP | 20,082,580 |

After defining the dataset, 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; for the current validation tests, three algorithms

needed to be selected. The best three algorithms were selected from the following range: decision tree classifier, random forest classifier, K-nearest neighbors, simple logistics, and support vector machine. These five algorithms were compared in terms of accuracy, precision and execution time when using the following four subsets of records from the CICDDoS2019 dataset:

- Subset A—collected using 186,960 records categorized as DDoS_Portmap and 4734 as Benign. Oversampling over minority class was applied and resulted in a balanced dataset with 373,920 records (186,960 DDoS_Portmap and 186,960 Benign);
- Subset B—collected using 782,590 records categorized as DDoS_UDP and 1071 as Benign. Oversampling over minority class was applied and resulted in a balanced dataset with 1,565,180 records (782,590 DDoS_UDP and 782,590 Benign);
- Subset C—collected using 1,289,043 records categorized as DDoS_SNMP and 925 as Benign. Oversampling over minority class was applied and resulted in a balanced dataset with 2,578,086 records (1,289,043 DDoS_SNMP and 1,289,043 Benign);
- Subset M—collected using 1,059,153 records (125,032 DDoS_DNS, 54,490 DDoS_LDAP, 112,410 DDoS_MSSQL, 101,756 DDoS_NetBIOS, 17,740 DDoS_NTP, 128,951 DDoS_SNMP, 65,155 DDoS_SSDP, 39,551 DDoS_SYN, 300,367 DDoS_TFTP, 77,466 DDoS_UDP, 14,280 Portmap 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).

Before the subsets were trained, a feature selection was made based on the most important features. In parallel, to balance subsets A, B and C, 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. 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 2 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, B and C) presented an accuracy ranging from 86% to 93%. In turn, the results of subset M (records of multi services) presented a lower accuracy ranging from 79% to 81%. The accuracy results show that the best three algorithms are the random forest classifier, the simple logistics, and the decision tree classifier.

**Table 2.** Results of the accuracy of each algorithm for each subset.

| | Accuracy (%) | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Subset A** | **Subset B** | **Subset C** | **Subset M** | **Average** |
| Random Forest Classifier | 93.25 | 91.54 | 92.93 | 81.26 | 89.75 |
| Simple Logistics | 92.38 | 91.16 | 91.27 | 80.93 | 88.94 |
| Decision Tree Classifier | 91.48 | 91.22 | 92.51 | 80.56 | 88.94 |
| K-Nearest Neighbors | 91.87 | 90.91 | 92.57 | 79.27 | 88.66 |
| Support Vector Machine | 88.83 | 86.12 | 91.34 | 80.19 | 86.62 |

Table 3 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, B, and C) presented a precision ranging from 88% to 93%. In turn, the results of subset M (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 3.** Results of the precision of each subset for each tested algorithm.

| | Precision (%) | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Subset A** | **Subset B** | **Subset C** | **Subset M** | **Average** |
| Random Forest Classifier | 91.96 | 91.46 | 92.61 | 81.75 | 89.45 |
| Decision Tree Classifier | 90.31 | 91.88 | 92.83 | 80.54 | 88.89 |
| Simple Logistics | 91.36 | 91.45 | 90.78 | 80.35 | 88.49 |
| K-Nearest Neighbors | 90.93 | 89.52 | 91.82 | 79.79 | 88.02 |
| Support Vector Machine | 90.19 | 88.04 | 90.71 | 80.04 | 87.25 |

From the results presented in Tables 2 and 3, it is observed that the highest values regarding accuracy and precision were obtained for the datasets with records per service (Subset A to C).

Table 4 presents the elapsed runtime for each algorithm and per subset. To obtain these results, each algorithm and subset was trained using a workstation featuring an Intel® Core™ i5-9400 CPU @ 2.90 GHz, 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 4.** Elapsed runtime results for each algorithm and per subset.

| | Elapsed Runtime (s) | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Subset A** | **Subset B** | **Subset C** | **Subset M** | **Average** |
| Decision Tree Classifier | 2 | 5 | 16 | 4 | 6.75 |
| Simple Logistics | 124 | 37 | 18 | 238 | 104.25 |
| Random Forest Classifier | 161 | 782 | 1489 | 472 | 726.00 |
| K-Nearest Neighbors | 426 | 6333 | 19,069 | 2988 | 7204.00 |
| Support Vector Machine | 1034 | 17,919 | 47,896 | 1834 | 17,170.75 |

In order to test the performance of the proposed architecture, their components (the master, the classifiers, and the trainers) were implemented in Fed4FIRE+ with Ray to distribute tasks by the available resources (other federated testbed platforms and distributed tasks tools could be used in the case that they present similar specifications). From the preliminary tests regarding models accuracy, precision, and elapsed runtime, the decision tree classifier (DTC), the random forest classifier (RFC), and the simple logistics (SLog) 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 100,000 classified records. Five samples of the main CICDDoS2019 dataset were prepared and tested in two scenarios: "baseline" and "proposed architecture". The

samples were obtained from the "DDoS_TFTP" service, and each sample (Sample #0 to Sample #4) includes 200,000 random records, sequential in time (i.e., if the first sample is collected from 14 to 16 h, the following is from 16 to 20 h).

Sample #0 was used for the initial operation. It was trained in the trainers using the three algorithms selected, and generated three machine learning models. The accuracy and precision results for Sample #0 using the three models are presented in Table 5. 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.** 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 Figure 8. 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 the accuracy and precision (1). Each sample was sent, unclassified, to the master (2), and the classifiers categorized each sample; 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, with 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.

Table 6 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, the 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 6.** 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 |

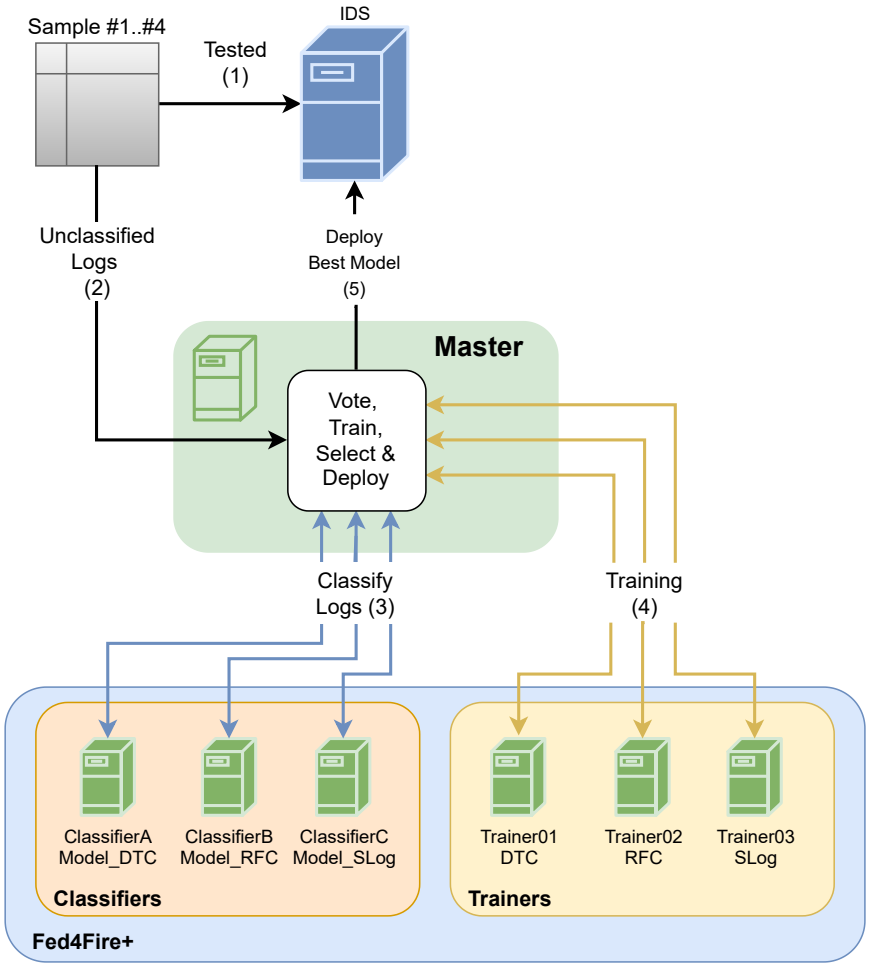


**Figure 8.** Proposed architecture scenario.

Tables 7 and 8 present, 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.

Figure 9 draws the results of Tables 7 and 8. 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.

**Table 7.** 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 8.** 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 |

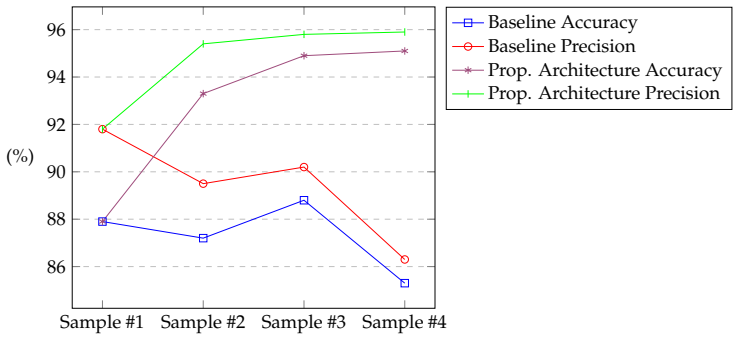


**Figure 9.** Results of accuracy and precision for the baseline and the proposed architecture scenarios.

## 5. Conclusions

An IDS assists system administrators by preventing and actuating on potential threats to systems and data. Anomaly-based IDSs 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 be trained with multiple algorithms.

In this paper, a centralized and vote-based architecture is proposed to generate classified datasets and improve the 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. In order to test the performance of the proposed architecture, its components were implemented in Fed4FIRE+ with Ray to distribute tasks by the available resources. 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 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. This best model can then be applied to the IDSs of each company, to improve their attack detection performance.

Future efforts may be developed to balance the trade-off between complexity and elapsed execution time against detection performance. Furthermore, the current proposal can be tested using deep learning models and against other types of attacks and datasets.

**Author Contributions:** Conceptualization, D.T., S.M. and P.P.; methodology, D.T., S.M. and P.P.; software, D.T.; validation, D.T., S.M. and P.P.; investigation, D.T.; writing—original draft preparation, D.T.; writing—review and editing, D.T., S.M. and P.P.; supervision, S.M. and P.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was partially supported by the Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF), within project "Cybers SeC IP" (NORTE-01-0145-FEDER-000044).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** No other data available from external sources.

**Acknowledgments:** This study was developed in the context of a project in the Master in Cybersecurity at the Instituto Politécnico de Viana do Castelo, Portugal.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## References

1. Grispos, G. Criminals: Cybercriminals. *Encycl. Secur. Emerg. Manag.* **2019**, 1–7. [CrossRef]
2. Truong, T.C.; Diep, Q.B.; Zelinka, I. Artificial Intelligence in the Cyber Domain: Offense and Defense. *Symmetry* **2020**, *12*, 410. [CrossRef]
3. Singh, M.D. Analysis of Host-Based and Network-Based Intrusion Detection System. *Comput. Netw. Inf. Secur.* **2014**, *8*, 41–47. [CrossRef]
4. Jyothsna, V.; Prasad, R.; Prasad, K.M. A review of anomaly based intrusion detection systems. *Int. J. Comput. Appl.* **2011**, *28*, 26–35. [CrossRef]
5. Fed4FIRE+. About Fed4FIRE+. 2022. Available online: https://www.fed4fire.eu/the-project/ (accessed on 10 January 2022).
6. Kumar Singh Gautam, R.; Doegar, E.A. 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, Noida, India, 11–12 January 2018; pp. 61–64. [CrossRef]
7. Tirumala, S.S.; Sathu, H.; Sarrafzadeh, A. Free and open source intrusion detection systems: A study. In Proceedings of the 2015 International Conference on Machine Learning and Cybernetics (ICMLC), Guangzhou, China, 12–15 July 2015; Voume 1, pp. 205–210. [CrossRef]
8. Hussein, S.M. Performance Evaluation of Intrusion Detection System Using Anomaly and Signature Based Algorithms to Reduction False Alarm Rate and Detect Unknown Attacks. In Proceedings of the 2016 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 15–17 December 2016; pp. 1064–1069. [CrossRef]
9. Ahmad, T.; Anwar, M.A.; Haque, M. *Machine Learning Techniques for Intrusion Detection*; IGI Global: Hershey, PA, USA, 2013; pp. 47–65. [CrossRef]
10. Khosravifar, B.; Bentahar, J. An Experience Improving Intrusion Detection Systems False Alarm Ratio by Using Honeypot. In Proceedings of the 22nd International Conference on Advanced Information Networking and Applications (AINA 2008), Gino-wan, Japan, 25–28 March 2008; pp. 997–1004. [CrossRef]
11. Venkatesan, R.; Devi, D.R.; Keerthana, R.; Kumar, A.A. A Novel Approach for Detecting Ddos Attack in H-IDS Using Association Rule. In Proceedings of the 2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA), Pondicherry, India, 6–7 July 2018; pp. 1–5. [CrossRef]
12. Teixeira, D.; Assunção, L.; Pereira, T.; Malta, S.; Pinto, P. OSSEC IDS Extension to Improve Log Analysis and Override False Positive or Negative Detections. *J. Sens. Actuator Netw.* **2019**, *8*, 46. [CrossRef]
13. Gao, X.; Shan, C.; Hu, C.; Niu, Z.; Liu, Z. An Adaptive Ensemble Machine Learning Model for Intrusion Detection. *IEEE Access* **2019**, *7*, 82512–82521. [CrossRef]
14. Panda, M.; Patra, M. Ensemble voting system for anomaly based network intrusion detection. *Full Pap. Int. J. Recent Trends Eng.* **2009**, *2*, 8.
15. Raykar, V.C.; Yu, S.; Zhao, L.H.; Jerebko, A.; Florin, C.; Valadez, G.H.; Bogoni, L.; Moy, L. 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, Montreal, QC, Canada, 14–18 June 2009; pp. 889–896. [CrossRef]
16. Mauro, M.D.; Sarno, C.D. Improving SIEM capabilities through an enhanced probe for encrypted Skype traffic detection. *J. Inf. Secur. Appl.* **2018**, *38*, 85–95. [CrossRef]
17. Mahfouz, A.; Abuhussein, A.; Venugopal, D.; Shiva, S. Ensemble Classifiers for Network Intrusion Detection Using a Novel Network Attack Dataset. *Future Internet* **2020**, *12*, 180. [CrossRef]

18. Haghighat, M.H.; Li, J. Intrusion detection system using voting-based neural network. *Tsinghua Sci. Technol.* **2021**, *26*, 484–495. [CrossRef]
19. Gulla, K.K.; Viswanath, P.; Veluru, S.B.; Kumar, R.R. Machine learning based intrusion detection techniques. In *Handbook of Computer Networks and Cyber Security: Principles and Paradigms*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 873–888. [CrossRef]
20. Haripriya, L.; Jabbar, M.A. Role of Machine Learning in Intrusion Detection System: Review. In Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018, Coimbatore, India, 29–31 March 2018; pp. 925–929. [CrossRef]
21. Shin, I.; Choi, Y.; Kwon, T.; Lee, H.; Song, J. Platform design and implementation for flexible data processing and building ML models of IDS alerts. In Proceedings of the 2019 14th Asia Joint Conference on Information Security, AsiaJCIS 2019, Kobe, Japan, 1–2 August 2019; pp. 64–71. [CrossRef]
22. Vikram, A.; Mohana. Anomaly detection in Network Traffic Using Unsupervised Machine learning Approach. In Proceedings of the 2020 5th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 10–12 June 2020; pp. 476–479. [CrossRef]
23. Anthi, E.; Williams, L.; Słowińska, M.; Theodorakopoulos, G.; Burnap, P. A Supervised Intrusion Detection System for Smart Home IoT Devices. *IEEE Internet Things J.* **2019**, *6*, 9042–9053. [CrossRef]
24. Taher, K.A.; Mohammed Yasin Jisan, B.; Rahman, M.M. Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection. In Proceedings of the 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST), Dhaka, Bangladesh, 10–12 January 2019; pp. 643–646. [CrossRef]
25. Ahanger, A.S.; Khan, S.M.; Masoodi, F. An Effective Intrusion Detection System using Supervised Machine Learning Techniques. In Proceedings of the 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 8–10 April 2021; pp. 1639–1644. [CrossRef]
26. Rani, D.; Kaushal, N.C. Supervised Machine Learning Based Network Intrusion Detection System for Internet of Things. In Proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 1–3 July 2020; pp. 1–7. [CrossRef]
27. Shi, G.; He, G. Collaborative Multi-agent Reinforcement Learning for Intrusion Detection. In Proceedings of the 2021 7th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC), Beijing, China, 17–19 November 2021; pp. 245–249. [CrossRef]
28. Latif, S.; Huma, Z.E.; Jamal, S.S.; Ahmed, F.; Ahmad, J.; Zahid, A.; Dashtipour, K.; Umar Aftab, M.; Ahmad, M.; Abbasi, Q.H. Intrusion Detection Framework for the Internet of Things using a Dense Random Neural Network. *IEEE Trans. Ind. Informatics* **2021**, 1. [CrossRef]
29. Kunal; Dua, M. Machine Learning Approach to IDS: A Comprehensive Review. In Proceedings of the 3rd International Conference on Electronics and Communication and Aerospace Technology, ICECA 2019, Coimbatore, India, 12–14 June 2019; pp. 117–121. [CrossRef]
30. Mo, W.; Gutterman, C.L.; Li, Y.; Zhu, S.; Zussman, G.; Kilper, D.C. Deep-neural-network-based wavelength selection and switching in ROADM systems. *J. Opt. Commun. Netw.* **2018**, *10*, D1–D11. [CrossRef]
31. Verbraeken, J.; Wolting, M.; Katzy, J.; Kloppenburg, J.; Verbelen, T.; Rellermeyer, J.S. A Survey on Distributed Machine Learning. *ACM Comput. Surv.* **2020**, *53*, 1–33. [CrossRef]
32. Global Environment for Network Innovations (GENI). What Is GENI? 2022. Available online: https://www.geni.net/about-geni/what-is-geni/ (accessed on 10 January 2022).
33. Smart Applications on Virtual Infrastructure (SAVI), 2022. Available online: https://www.savinetwork.ca/ (accessed on 10 January 2022).
34. Kim, T.; Cha, Y.; Shin, B.; Cha, B. Survey and Performance Test of Python-Based Libraries for Parallel Processing. In Proceedings of the 9th International Conference on Smart Media and Applications. Association for Computing Machinery, New York, NY, USA, 23 August 2020; pp. 154–157. [CrossRef]
35. Ray Team. What Is Ray? 2021. Available online: https://docs.ray.io/en/master/ (accessed on 12 January 2022).
36. Using IPython for Parallel Computing. Available online: https://ipython.org/ipython-doc/3/parallel/ (accessed on 12 January 2022).
37. dispy: Distributed and Parallel Computing with/for Python—Dispy 4.12.0 Documentation. Available online: https://dispy.org/ (accessed on 12 January 2022).
38. Pandaral lel. 2021. Available online: https://github.com/nalepae/pandarallel/tree/v1.5.4 (accessed on 12 January 2022).
39. Dask. Dask—Documentation. 2022. Available online: https://docs.dask.org/en/stable/ (accessed on 12 January 2022).
40. Joblib. Joblib: Running Python Functions as Pipeline Jobs. 2022. Available online: https://joblib.readthedocs.io/en/latest/ (accessed on 12 January 2022).
41. Moritz, P.; Nishihara, R.; Wang, S.; Tumanov, A.; Liaw, R.; Liang, E.; Elibol, M.; Yang, Z.; Paul, W.; Jordan, M.I.; et al. Ray: A Distributed Framework for Emerging AI Applications. In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). USENIX Association, Carlsbad, CA, USA, 8–10 October 2018; pp. 561–577.
42. Fabbiani, E.; Vidal, P.; Massobrio, R.; Nesmachnow, S. Distributed Big Data Analysis for Mobility Estimation in Intelligent Transportation Systems. *Transp. Rev.* **2019**, *6*, 795–818. [CrossRef]

43. Nsl-kdd Dataset. 2014. Available online: https://www.unb.ca/cic/datasets/nsl.html (accessedon 21 December 2021).
44. KDD Cup 1999. 1999. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 21 December 2021).
45. The UNSW-NB15 Dataset Description. 2015. Available online: https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed on 21 December 2021).
46. Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In Proceedings of the International Carnahan Conference on Security Technology, Chennai, India, 1–3 October 2019. [CrossRef]