F-NIDS – A Network Intrusion Detection System based on Federated Learning

Jonathas A. de Oliveira[a,*], Vinícius P. Gonçalves[a], Rodolfo I. Meneguette[b], Rafael T. de Sousa Jr[a], Daniel L. Guidoni[c], Geraldo P. Rocha Filho[d]

[a] *University of Brasilía, Department of Electrical Engineering, Brasilía - DF, Brazil*
[b] *University of São Paulo, Department of Computing System, São Carlos, SP, Brazil*
[c] *Department of Computer Science, Federal University of Ouro Preto, Brazil*
[d] *Department of Computer Science State University of Southwest Bahia, Candeias, BA, Brazil*

Abstract

The emergence of IoT networks has brought about new challenges in scalability and security due to the large number of connections and high data transfer rates in these networks. While efforts have been made in recent years to address these challenges, there are still unresolved questions related to data privacy and scalability in distributed IoT scenarios. In this work, we propose the F-NIDS, an intrusion detector that utilizes federated artificial intelligence and differential privacy techniques, along with asynchronous communication between system entities, to address scalability and data confidentiality concerns. The architecture of F-NIDS is designed to be adaptable for usage in cloud or fog IoT environments. Results from our experiments have shown that the confidential detection model employed in F-NIDS maintains satisfactory performance metrics and is capable of predicting and determining the nature of attacks when they occur. In order to determine optimal parameters that strike a balance between data privacy and classification performance, three strategies were employed, each evaluated for its corresponding robustness performance. Firstly, models were trained with varying Gaussian noise values, and subjected to membership inference black box rule-based attacks. Secondly, regular membership inference black box attacks were performed, utilizing different stolen samples with varying sizes to determine the maximum amount of data that could be securely stored on the detection agents for training tasks. Lastly, the robustness of the trained models was evaluated against a model inversion attack, and the results were compared through graphical comparisons. Based on these evaluations, Gaussian noise level and sample size values of 21 were obtained for each detection agent in the system, with sample sizes ranging from 10K to 25K.

Keywords: Network Intrusion Detection System, Federated Learning, Asynchronous Messaging, Security Systems, Distributed Computing, Pub/Sub mechanism

*Corresponding author
Email addresses: `jonathas.oliveira@aluno.unb.br` (Jonathas A. de Oliveira), `vpgvinicius@unb.br`

1. Introduction

The past decade we have seen a considerable increase in the interconnection between humans, machines, and services. This has resulted in the communication paradigm of the IoT - Internet of Things [1, 2]. Concerns, such as scalability, latency, and information privacy, have been raised within the context of IoT [3]. Decentralized architectures may bring some solutions to these issues, offering higher availability and superior scalability [4, 5]. On the security front, the most popular strategies used in IoT networks comprise the use of NIDS (Network Intrusion Detection Systems). These systems are responsible for determining and alerting whether a particular activity is normal or malicious on the network [6]. By the other hand, conventional NIDS techniques may be less effective in the IoT context, due to their dynamism [7].

A NIDS is built to provide continuous monitoring and detection during the life cycle of computer networks [8]. However, due to the dynamic nature of an IoT environment, whose resources are often limited and may contain heterogeneous devices of very high connectivity, more common NIDS techniques may be less effective for intrusion detection systems [7]. Therefore, NIDS works under more challenging and restrictive circumstances when used in this environment. On the occasion of the excellent results, researchers have adopted ML approach (Machine Learning) for NIDS development to improve cyberattack detection [6]. On the other hand, the authors discussed in [9] the issue of privacy in ML models still remains a challenge, especially in a decentralized architectures. This is still a promising field for the development of alternatives that increase the security and confidentiality of the training data and models, while maintaining the scalability and resilience that decentralized architectures can offer.

Although there is a vast amount of work on the issue of privacy in the context of ML, there are still few applications in a real-life scenario of use [10] [11]. In the aspect of decentralization, FL (Federated Learning) is a recently proposed paradigm to enable the distribution of ML tasks with greater privacy preservation of training data and this technique has demonstrated a wide range of applications, especially where confidentiality is an important aspect [12].

In this context, FL is an ML technique that allows creating decentralized learning architectures. It enables integration with DP (Differential Privacy) mechanisms, mitigating some of the limitations of NIDS already mentioned. In this technique, many clients collaboratively train a model separately, under the orchestration of a central server. FL enables the training data to remain decentralized and restricted at the source without sharing it [13]. In this technique, the central server can receive only the weights and gradients of the models trained by clients [12]. While keeping isolated training data on clients may in theory increase confidentiality, there still remains the risk from inference of information from the trained models [14]. DP ensures that any version of a statistical dataset remains equally

(Vinícius P. Gonçalves), meneguette@icmc.usp.br (Rodolfo I. Meneguette), desousa@unb.br (Rafael T. de Sousa Jr), guidoni@ufop.edu.br (Daniel L. Guidoni), geraldo.rocha@uesb.edu.br (Geraldo P. Rocha Filho)

credible regardless of whether it contains a particular item [15], minimizing the possibility of inference of sensitive individual information, but maintaining the statistical properties of the dataset.

Different approaches for decentralized IDSs have been proposed with advances in the field [16, 17]. These solutions address the use of multi-agents with machine learning and focus primarily on performance and scalability issues. Other decentralized IDS proposals, which explore with more emphasis the issue of data privacy and trustworthiness, have been investigated in [18, 19]. Although these are architectures focused on data privacy, the former work admits a central point of vulnerability. The use of federated learning has also been under evaluation for building intrusion detector [20] [21], however they were not focused on the specific requirements of IoT networks.

The goal of this work is to propose the F-NIDS, a decentralized intrusion detection system in order to overcome the mentioned limitation of literature works. The system is based on FL, and aims to offer a distributed architecture without the need to exchange client data, as it dispenses with the transmission of individual client information. And, to ensure model protection, it uses DP techniques, enabling a more confidential transmission of models between clients. F-NIDS contains the following main features: (i) presents a high accuracy of data traffic classification in a distributed IoT scenario; (ii) endowed with a decentralized architecture that allows it to scale quickly; and (iii) guarantees higher confidentiality of both training data and trained models. F-NIDS was evaluated and compared with three different approaches, and it presents similar results and the proposed solution is able to predict and determine the nature of attacks.

The remainder of the article is organized as follows. Section II presents the related articles. Section III describes the F-NIDS. Section IV presents the results through performance evaluation. Section V concludes the paper with the conclusion and future work.

2. Related works

Centralized NIDS solutions have been recently developed aiming to extend the level of accuracy and performance of intrusion detections [22] [23] [24]. Specifically, in [24] they propose the use of convolutional neural networks (ConvNet) in intrusion detection systems. In [22], a deep neural network is defined with stacking of asymmetric auto encoders, combined with an output layer in SVM - Support Vector Machines to achieve better levels of accuracy in attack classifications, a technique called (S-NDAE). However, besides the lack of mechanisms to guarantee the confidentiality of the resulting models, the cited works have limitations regarding their scalability to serve highly distributed networks, as explored in this research [6].

In the context of decentralized NIDS's, different solutions have been proposed [16] [17]. These solutions address the use of multi-agents with machine learning and focus mainly on performance and scalability issues. The use of FL has also been evaluated for building intrusion detectors [20], however they are limited to analyzing data coming from conventional networks, without the focus on the specific characteristics of IoT networks and the confidentiality of the models. However, besides the lack of mechanisms to guarantee the confidentiality of the resulting models, the cited works have limitations mainly to serve highly distributed networks, as explored on [6].

Other decentralized NIDS solutions, which explore the issue of data privacy and reliability with more emphasis, have been investigated in [18] [19]. [18] proposes SP-CIDS, an ML solution for distributed NIDS aimed at serving autonomous vehicle networks, by applying PD techniques to the training data. [19] develops a distributed NIDS for healthcare systems. This proposal uses generative neural networks and auto-encoders to protect the confidentiality of the models. For model aggregation and transmission [18] adopt DMS (Distributed Machine Learning) in model aggregation. Although these are architectures aimed at protecting training data, both solutions admit a central point of vulnerability, allowing the transfer of raw data to between a central agent, and are therefore ineffective for privacy protection [18]qian2020. Furthermore, both works use the assumption that generative neural networks inherently produce more confidential models for storage and transmission, however as demonstrated in [25], such ML techniques can also be vulnerable to confidentiality violations, specifically inference attacks on the training data.

In [26], a federated NIDS is proposed that privileges confidentiality, using DP techniques. This is the solution that most closely resembles F-NIDS. The authors adopted the `Fed+` model aggregation algorithm. This algorithm is said to provide greater accuracy of the overall model, reducing the effects of noise caused by DP, and mitigating the losses caused by the aggregation itself. Although it proposes a decentralized NIDS, the scenario only addresses a subset of IoT applications (i.e., industrial applications), and does not evaluate its generalization to other IoT applications, such as, for example, autonomous cars, smart cities, and in datacenters in general. Furthermore, some limitations of the clustering algorithm are raised, regarding the issue of the agents' customization capability and also in the robustness of the generated models [27]. It is worth noting that [26] used, in the evaluation of the results, a too small amount of clients, which may have impacted the numbers obtained, being a strategy that may be inadequate to test a real federated learning scenario, in which a few tens or even millions of clients are predicted. A more robust study with larger numbers of clients can be performed to validate NIDS using federated learning [28].

Some works already was made proposing systems that use pub/sub mechanisms to provide communication between system entities. Thus, validating the scalability and availability provided by such technique. In [29] was proposed a framework, using this technique, which was capable of improving the processing speed of large-capacity data ensuring more stability and resilience over adverse conditions such as restricted bandwidth on IoT cloud platforms. For the other hand, [30] proposed a communication solution which applies pub/sub technique in conjunction with NDN (Named Data Networking) and leveraging the availability and scalability of the data exchange between entities. The evaluations performed in [31] reached the conclusion that exists open-source tools for pub/sub, on both IoT and cloud environments, which are error-tolerant and support live-stream processing operations, while can operate well in cluster. Thus, maintaining fault-tolerance, supporting heavy workloads, so they have proving to be scalable solutions.

The evaluation of these works allowed us to conclude that there are still open problems in this field. Among these problems is the absence of a distributed architectural proposal capable of meeting various workloads. The other problem, and no less important, concerns the confidentiality of information, so there is room for a proposal that addresses with greater emphasis the issue of data privacy, in addition to collaboratively trained models. In this sense, F-NIDS was proposed aiming to deliver solutions for the scalablity and availability issues, on IoT and cloud contexts, by using multiple distributed detection instances exchanging data via FL mechanism and communicating with the clients through pub/sub mechanism. The problem of privacy vulnerability coming from hypothetical data leakage in the detection agents is solved by spreading less data across higher number of detection agents, and transmitting just the trained weights, using federated learning. In case of privacy vulnerability, coming from a hypothetical model theft on a given instance, can be reduced by adopting differential privacy mechanisms during the training step performed by the detection agents.

3. F-NIDS - Federated Network Intrusion Detection System

This section presents the F-NIDS, a federated intrusion detection system coupled with DP and *publish/subscribe* mechanisms. In addition to protecting the confidentiality of a model's training data, it also deals with the confidentiality of the model itself against attacks using membership inference techniques in a distributed training and detection scenario. Initially, an overview of the system and its operation are presented, along with the tools used for its modeling.

3.1. Overview

The overall architecture of F-NIDS is illustrated in Figure 1. In the figure, the central agent (label CA) generates the initial global weights (label G), propagating them to the other members of the system. The CA has the function of aggregating weights and producing a global model by means of a local weight aggregation technique, and also performs the orchestration, propagating the weights obtained in this aggregation. The tasks of detection, reception of the global weights and training of a local model is done by the Detection Agents (label DA). These DA's, besides training a local model using an individual dataset, receive detection requests through an auxiliary sub-system called *Message Broker (MB)* , which receives messages, stores and routes them to one or more recipients asynchronously. Clients send detection requests through this mechanism, publishing them in internal MB queues, thus allowing scalability and reliability gains. To this end, the packet capture task is done by the clients themselves individually. To provide greater robustness against inference attacks on the models, each DA comes equipped with an additional security layer using DP. Thus, the DA produces a global model that inherits some DP properties from the other local models. This architecture is based on three mechanisms: (i) decentralized training mechanism using FL; (ii) training mechanism with DP and (iii) decentralized and distributed detection mechanism.

Another aspect is that F-NIDS only transacts the synaptic weights of the models. This is to minimize the transfer of data that may be vulnerable to confidentiality violations and compromise the available bandwidth. To provide greater robustness against inference attacks on the models, during the process of obtaining the local weights by the DA's, an additional security layer was included using DP, aimed at protecting the privacy of the obtained models. Thus, each DA trains its own classifier with a certain degree of noise in order to minimize vulnerability to inference attacks. In aggregation, the CA produces a global model that inherits some differential privacy properties from the other agents.

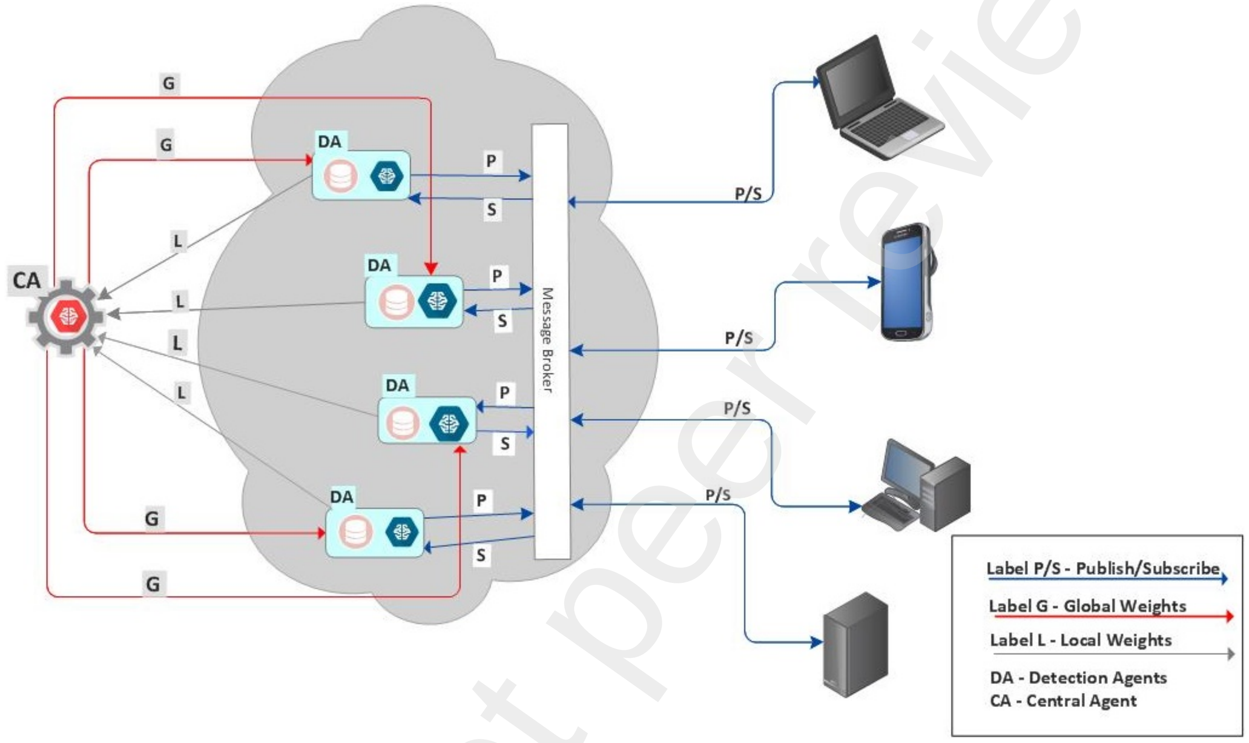


Figure 1: General F-NIDS architecture.

3.2. *Privacy federated learning mechanism*

Although F-NIDS is distributed, the central agent generates the initial model with random parameters and performs the orchestration of model training and transmission of these parameters. This orchestration is performed through variables called federated hyperparameters. These hyperparameters (arranged in Table 1) define the settings used in the trained classifiers. When all the conditions defined in the hyperparameters are met, a federated round is started in which the weight aggregation algorithm, called `FedAvg` (Federated Averaging) is applied. The federated round concludes when the propagation of the federated model parameters to the detection agents is finished.

In the `FedAvg` [13] the agent $k_t \in K$ (with $n_k = |\mathcal{P}_k|$, where $\mathcal{P}_k$ is the indices of the dataset contained in agent k_t considering $C = 1$ as the complete dataset, and η is the learning rate) computes the gradient vector $g_k = \nabla F_k(w_t)$ corresponding to the local training of model w_t. Then, the agents themselves are assigned the task of updating the weights locally by $k_t \leftarrow k_t - \eta \nabla F_k(k_t)$ several times before the aggregation step, which is still performed by

Table 1: List of hyperparameters used in model training.

Hiperparameter	**Default value**
Neurons in the hidden layer	160
Learning rate	0.02
Epochs	10
Rounds	10
Minibatch size	1000
Validation set	20%
DP-SGD - Norm L_2	1.5
DP-SGD - Noise σ	0.5
FL – Minimum fraction of training DA's	0.1.
FL – Minimum fraction of assessment DA's	0.1
FL – Minimum training DA's	10
FL – Minimum available DA's	75

the CA. In this strategy, the computational cost is controlled by three hyperparameters: C, being the fraction of agents performing the computations in each round; E the number of epochs of each agent in its local dataset; and B the size in the *minibatch* used by each client. Thus the `FedAvg` has the value of $B = \infty$ (minibatch size equal to local dataset size) and $E = 1$. The CA aggregates these gradients and applies the $w_{t+1} \leftarrow w_t - \eta \nabla f(w_t)$, where $\nabla f(w_t) = \sum_{k=1}^{K} \frac{n_k}{n} g_{t+1}^{k}$.

Algorithm 1 describes this process of training the F-NIDS classification models. It consists of two steps, the global step (from lines 1 to 9) and the local step (from lines 10 to 16). In row 1 a vector of global weights is initialized. In the next line it iterates over each round and internally calculates the number of clients that will participate in training for a given round. The fifth line iterates over some of the DA's registered in F-NIDS, passing as argument the weights of the global model (line 6) and obtaining a vector of local weights for each agent. After obtaining the local weights of a given agent, the result is indexed into a vector containing each of these weights. In line 8 the actual aggregation of all the weights of the locally trained models, for each DA, is done. Then the global weights are updated to be used in the next round. The local stage starts on line 10 and onwards, with the training of the local models on the agents. The step starts with local training data being split into a B set of mini-batches. In lines 11 and 12, it iterates over each of the i epochs of the agents. For each of these epochs, all the $b \in B$ mini-batches are used as arguments to compute the local weights of the current agent (line 13), in addition to the global weights and the learning rate of the agent in question. The local step ends when a set of local weights w of the model are obtained and passed on to the central agent.

3.3. Classifier's Differential Privacy Mechanism

In order to ensure the highest confidentiality of the trained models, the algorithm `DP-SGD` (*Differentially Private - Stochastic Gradient Descent*) is used. This approach is the classic version of the SGD model optimization algorithm, but with the inclusion of PD. This

Algorithm 1 `FederatedAveraging`. K agents are indexed by k; B is the size of the local minibatch, E is the number of local epochs, and η is the learning rate.

```
    CentralAgent()                                   ▷ //Run on central agent
 1: initialize $w_0$
 2: for each round t = 1, 2, ... do
 3:     $m \leftarrow \max(C \cdot K, 1)$
 4:     $S_t \leftarrow$(random set of $m$ clients)
 5:     for each client $k \in S_t$ in parallel do
 6:         $w_{t+1}^k \leftarrow$DetectionAgentUpdates$(k, w_t)$
 7:     end for
 8:     $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
 9: end for

    DetectionAgentUpdates($k, w$):                   ▷ //Run on detection agent k
10: $\mathcal{B} \leftarrow$ ( split $\mathcal{P}_k$ into batches of size B)
11: for each local epoch $i$ from 1 to $E$ do
12:     for batch $b \in \mathcal{B}$ do
13:         $w \leftarrow w - \eta \nabla \ell(w; b)$
14:     end for
15: end for
16: return $w$ to central agent
```

algorithm limits the sensitivity of each gradient. It achieves this result by *clipping* and noises the gradient during training, aiming to amplify and monitor the privacy cost [32]. Let $clip_c : g_t(x_i) \in \mathbb{R}^p \rightarrow g_t(x_i)/max(1, \frac{\|g_t(x_i)\|_2}{C}) \in \mathbb{R}^p$ the clipping function applied over the input values such that the result has the maximum ℓ_2 norm of C. Thus, the update step by the DP-SGD algorithm is given by: $w^{(t+1)} = w^{(t)} - \eta_t\{\frac{1}{B}\sum_{i\in\mathbb{B}_t} clip_c(\nabla_{w_t}\mathcal{L}(w_t, x_i)) + \delta\}$ and $\delta \sim \mathcal{N}(0, \sigma^2C^2I)$ is the random variable corresponding to Gaussian DP and σ^2 the noise's standard deviation [33].

The complete pseudocode of `DP-SGD` is presented in Algorithm 2. Line 1 contains the list of hyperparameters used in training which are the training examples (i.e., learning rate, σ noise scale, L size, and the norm bound C). In line 2 a set of initial weights are initialized randomly. From line 3 the algorithm iterates over each epoch. In line 4, at each epoch a sample L_t is selected and for each subset i of L_t it computes the gradient vector $g_t(x_i)$. In line 8 the *clipping* of the obtained gradient vector is done, then adding the Gaussian noise in line 9 and finally adjusting the synaptic weights of the model as a function of the obtained gradient and the learning rate.

Algorithm 2 `DP-SGD`. Differentially private SGD

```
1:  Input: Samples $\{x_1, ..., x_N\}$, loss function $\mathcal{L}(w) = \frac{1}{N}\sum_i \mathcal{L}(w, x_i)$. Parameters: learning
    rate $\eta_t$, noise $\sigma$, minibatch size $L$, grandient norm bound $C$ and T is the epoch quantity.
2:  Random initialization $w_0$
3:  for $t \in [T]$ do
4:      Take a subsample $L_t$ with probability distribution $\frac{L}{N}$
5:      for $i \in L_t$ do                                          ▷ //Gets gradient
6:          $g_t(x_i) \leftarrow \nabla_{w_t}\mathcal{L}(w_t, x_i)$
7:      end for
8:      $\bar{g}_t(x_i) \leftarrow clip_c(g_t(x_i))$                 ▷ //Clip gradient
9:      $\tilde{g}_t \leftarrow \frac{1}{L}\sum_i(\bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2C^2I))$   ▷ //Gradient noise step
10:     $w_{t+1} \leftarrow w_t - \eta_t\tilde{g}_t$                 ▷ //Adjust weights
11: end for
12: Return $w_T$.
```

3.4. Detection mechanism

F-NIDS splits the intrusion detection process into three steps: Capture, classification, and countermeasures. Capture is done directly by clients, using an internal capture mechanism. Detection is performed by the DA's, by publishing and subscribing to MB queues. Countermeasures can be performed jointly by the agents and the devices based on their own rules, allowing each of the system members to implement its individual repudiation policy against malicious agents.

The proposed F-NIDS interaction between a client, interested in detecting a particular packet, and DA is illustrated in Figure 2. The client starts the process by capturing a packet and checking whether the sender is on the repudiation list. If the sender is not previously blocked, the client sends a classification request to the DA. It is important to point out that the client does not know which DA will be responsible for detecting the intrusion, nor its location, because the MB decouples the parties. However, the MB guarantees that the communication will have an asynchronous response, through the *publish/subscribe* mechanism. The classifier, upon predicting it as benign, notifies the interested client. If the packet is categorized as malicious, the F-NIDS will notify the initial client and issue an alert to all other clients that subscribed in the MB's alert queue. This notification contains the probability of the classification made, which class of attack was detected, and the source that issued the packet. The client, upon receiving attack notifications or an alert, includes the source in the repudiation list and terminates any connections to the sender of the malicious packet.

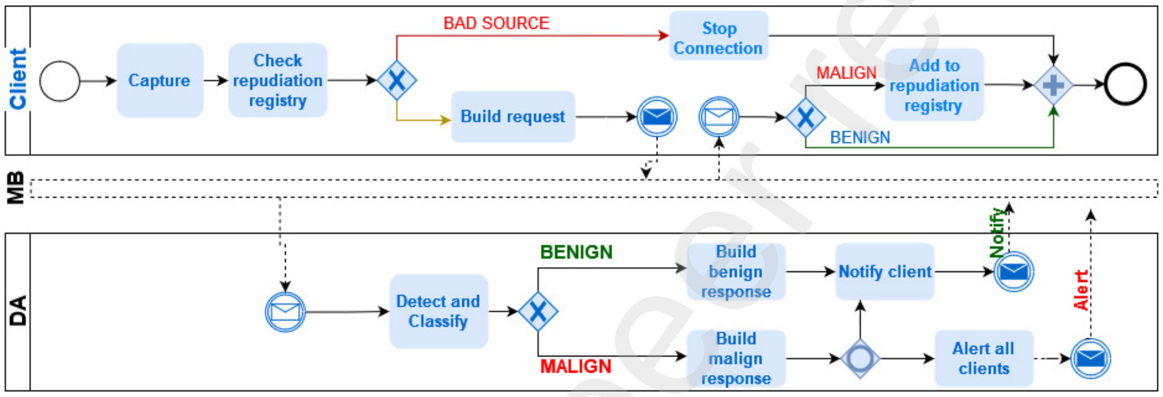


Figure 2: Basic operation between a certain DA and an client.

Figure 3 illustrates the communication model between all available DA's and the clients interested on detection intrusion. For the detection of an intrusion, a client c publishes a message m, formatted from a captured packet, by invoking the operation $pub(m)$. Upon performing benign classification of a message, the interested client is notified with a reply message r, via a $notify(c, r)$ operation. Besides detecting an intrusion, the agent publishes an alert message addressed to the interested client and also to all other clients, via the $alert(r, t)$ operation, where t is both the classification result and the alert type. A client subscribes to the alert results via the $sub(t)$ operation, t its optional, so if t is entered, the client subscribes only to the alert type entered, otherwise it will subscribe to alerts of all types. In the middle, a message broker cluster handles the messages, ensuring that only one of the detection agents obtains and processes it. In case of one instance of broker get unavailable, the communication continues to flow normally, because the cluster is capable to handle such event properly. It is important to note that the repudiation list is maintained individually by each client and this configuration is thought to allow each of these agents to implement their individual policy against malicious agents.

Each alert a or notification message m issued by the devices follows the IDEA (Intrusion Detection Extensible Alert) message format standard formulated by [34], a communication format that uses JSON (Javascript Object Notation) notation and which is based on the IDMEF (Intrusion Detection Message Exchange Format) proposed by [35]. In this standard,

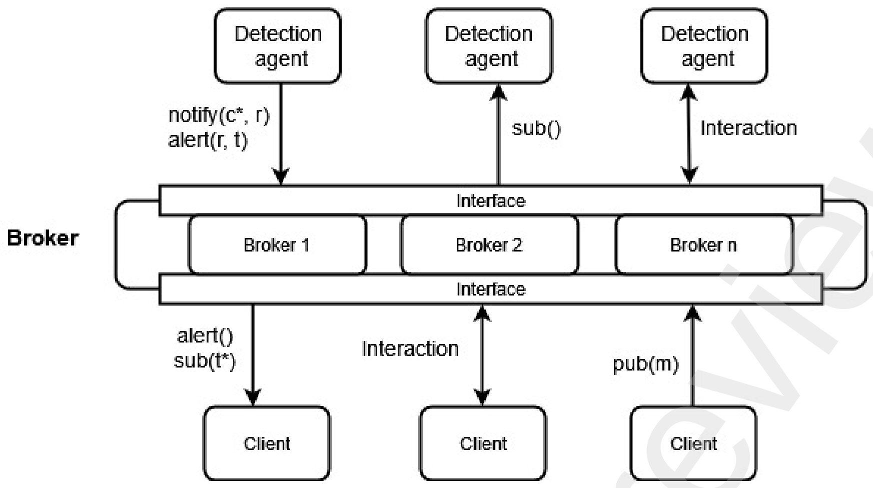


Figure 3: F-NIDS communication model.

all messages sent by detection agents need to have only one classification and be provided with a set of attributes with their respective types that identify the source, recipient and time tracking of a packet. The fundamental attributes are: AnalyzerID, wich is the identifier of the agent that made the alert; CreateTime is the date the message was generated by the sender of the packet; DetectTime and AnalyzerTime are the times relative to the date the packet was sent for analysis and the time it was analyzed by one of the detection agents, respectively. The sender IP addresses and the target of the packet are represented by the Source and Target attributes.

4. Methodology and results

In this section, the performance evaluation of F-NIDS is presented. F-NIDS is evaluated by comparing the performance metrics of multi-class and binary predictions against the other three methods studied. This strategy aims to verify that the F-NIDS is viable in terms of useful attack detection capability, despite applying two algorithms that penalize accuracy. These two algorithms are FedAvg and DP and they, with very low hyperparameter values, have a minor impact on the observed prediction metrics.

The performance of the methods was evaluated using, in addition to accuracy, *precision* and *recall*. Precision (P_c) is described in Equation 1 and is the measure of the classifier's performance in detecting a given class c, where (TP_c) is the examples correctly classified as being c and (FP_c) are the examples incorrectly classified as being of class c. Recall R_c is the ratio described in Equation 2, where FN_c are the examples incorrectly classified as being of class other than class c. Both metrics are computed on a class-by-class basis.

$$P_c = \frac{TP_c}{TP_c + FP_c} \tag{1}$$

$$R_c = \frac{TP_c}{TP_c + FN_c} \tag{2}$$

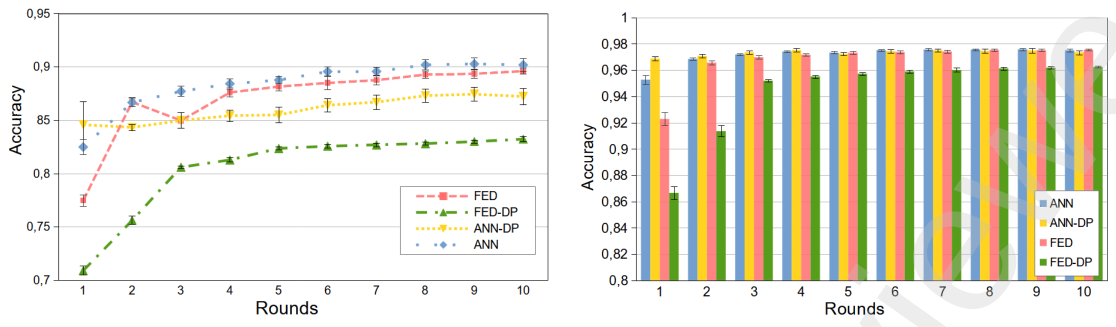
Four classifier training methods were evaluated: The centralized method named ANN; the centralized with DP named (ANN-DP); the training method applying only the federated algorithm (FED) and the federated method with DP (F-NIDS). Ten individual models were trained for each method, with each model evaluated on a random fraction of 10% of the test dataset. In order to obtain the overall multi-class and binary performance across the four methods, the overall accuracy will be analyzed over the training rounds In (Section 4.1). This strategy aims to observe at which round each method reaches relative levels of convergence. The section 4.2 deals with the evaluation of the obtained accuracy and *recall* results, aiming to verify the classification performance in each of the classes individually.

The dataset used is based on the NF-ToN-IoT-v2 [1]. This data was produced from the `.pcap` files of the ToN-IoT dataset and then processed to generate data in the standard NetFlow tool [2] [36], where the produced database has 43 relevant *features*. The sender and receiver addresses information has been removed and the resulting dataset is unbalanced and has 2.5×10^6 examples extracted randomly from the original data, which is approximately 14.75% of the volume of data contained in NF-ToN-IoT-v2. Considering these data, 80% was separated for the training set and 20% was allocated to the test set. The experiment uses the libraries Tensorflow [3] for neural network training, TF Privacy [4] for differential privacy, and Flower [5] for federated learning. Table presents the list of hyperparameters used in the training model. The hyperparameters of the multilayer perceptron were obtained using the *hyperparameter tuning* method, the size of the minibatch was the largest supported by the GPU used (NVIDIA V100 with 16GB of GDDR5). The hyperparameters related to federated learning were chosen taking into account the available CPU and RAM memory (8 cores and 24GB RAM). Other values of hyperparameters were tested, but with a lower classifier performance in terms of accuracy and convergence.

4.1. F-NIDS accuracy results

The Figure 4 presents the accuracy results obtained in the four methods under study (ANN, ANN-DP, FED and F-NIDS), as a function of the training rounds. Comparing the results in Figure 4a, the ANN method has on average 7.7% more accuracy than the F-NIDS method. When comparing the accuracy of the ANN method and the others, the differences were 0.06% over FED and 3.0% over ANN-DP, respectively. Such results are within expectations, since the `FedAvg` algorithm requires an additional cost in terms of classifier accuracy, since it needs to aggregate the weights obtained by local models trained with a significantly reduced number of examples. The model also have to consider the effect of the `DP-SGD` algorithm, which significantly impacts the accuracy of the classifier by including noises in the model. Based on the figures presented, it is possible to conclude that the multi-class performance differences are small between the F-NIDS method and all other methods studied.

[1]`https://staff.itee.uq.edu.au/marius/NIDS_datasets/`
[2]`https://www.cisco.com/c/pt_br/tech/quality-of-service-qos/netflow/index.html`
[3]`https://www.tensorflow.org`
[4]`https://www.tensorflow.org/responsible_ai/privacy/guide`
[5]`https://flower.dev/`



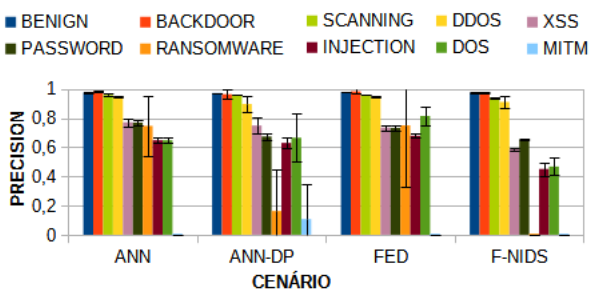
(a) Multicalss accuracy (b) Binary accuracy

Figure 4: Binary and multiclass accuracy per round

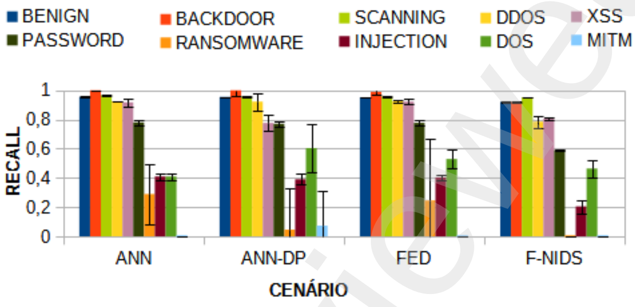
Figure 4b presents the binary accuracy results of the classifier during the ten training rounds. These results are obtained by grouping the attack classes into a single class. In this case, it can be seen that the detection accuracy between normal or malicious traffic shows even smaller differences. The impact on the accuracy of F-NIDS, relative to other methods, can be measured by the difference between the accuracies of the other methods and the accuracy of F-NIDS. In this case, it was found that the impact on the average binary accuracy of the ANN, ANN-DP, and FED methods, relative to the F-NIDS method is only 1.2%, 1.1%, and 1.3%, respectively. Therefore, no relevant performance changes between the four methods were observed within this context. These results allow us to conclude that the losses in binary accuracy, arising from the application of `FedAvg` and `DP-SGD` algorithms together, result in minor impacts of binary accuracy of the resulting model.

4.2. F-NIDS performance per class evaluation

In Figure 5, the accuracy and *recall* obtained on each class using the F-NIDS classification method is presented and compared to the ANN, ANN-DP, and FED methods. Figure 5a presents the accuracy, or proportion of hits among the predictions made, demonstrating the ability of the model to correctly classify a class. Figure 5b presents the *recall* of each class, separated by the four methods under study. This metric is the proportion of hits among the actual classes. The accuracy of *Benign, Backdoor, Scanning and DDOS* predictions show no significant differences of F-NIDS with the other methods. In the other classes (XSS, Password, Ransomware, Injection, DOS and MITM), noticeable and significant differences can be observed. It is worth noting the low accuracy obtained in the classification of *Ransomware and MITMT*. Apparently this behavior is associated with the amount of examples, which is less than the size of the mini-batch chosen as parameter L for the *DP-SGD* Algorithm. In the F-NIDS method, the DOS and Injection classes performed significantly below the other methods, although they had a number of training examples comparatively close to those of classes with better accuracy. These results allow us to conclude that the application of the `FedAvg` algorithm, in conjunction with `DP-SGD`, presented a small impacts on overall accuracy and multi-class recall.



(a) Precision



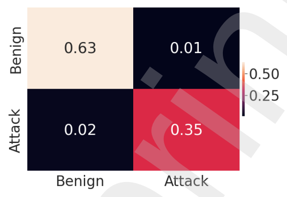
(b) Recall

Figure 5: Multi-class precision and recall results

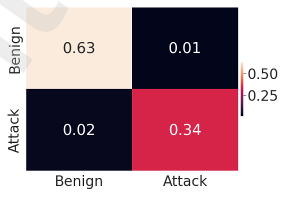
4.3. F-NIDS binary efficiency

Evaluating the results by class allows us to register the efficiency of the classifiers in detecting each class individually, but predicting whether a given packet is likely to be correctly classified as benign or malicious is a primary attribute for F-NIDS. Thus, the attack classes were grouped into just one class, named *Attack*, resulting in two possible classifications (i.e. benign; attack).

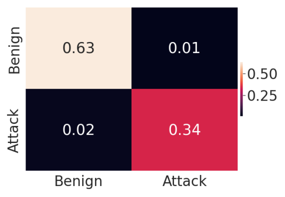
Figure 6 presents the confusion matrices for each of the methods, showing the intersection between the examples classified as benign or attack and their corresponding actual values. The F-NIDS method (Figure 6d) obtained a similar result to the others in the correct detection of benign traffic, which also contributed to maintain a reduced amount of false positives. In the detection of attacks, F-NIDS showed an amount of detection only slightly lower than the other methods, represented by Figures 6a, 6b and 6c. Furthermore, true attacks incorrectly classified as benign by the F-NIDS method had a small difference over the other methods. Such results demonstrate that the F-NIDS method is almost as efficient as the ANN (Figure 6a), ANN-DP (Figure 6b) and FED (Figure 6c) methods to detect benign packets and almost as efficient to correctly detect attacks as the other mentioned approaches. Table 2 numerically presents the results of the obtained metrics. It contains the averages of the ten observations and the errors represented by their standard deviation.



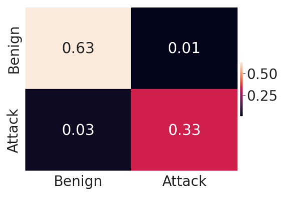
(a) ANN



(b) ANN-DP



(c) FED



(d) F-NIDS

Figure 6: Binary confusion matrices of the classifiers.

When comparing the effects of the DP-SGD algorithm on attack classification performance, by comparing the ANN method (Figure 6a) with ANN-DP (Figure 6b), it can be seen that this algorithm slightly influences the ability to detect true attacks. Considering all

Table 2: Evaluation of the methods' binary classification metrics. Each result is followed by its ± standard deviation.

Method	Precision	Recall	Accuracy
ANN	0.975 ± 0.005	0.955 ± 0.005	0.974 ± 0
ANN-DP	0.971 ± 0.002	0.954 ± 0.003	0.973 ± 0.001
FED	0.981 ± 0	0.980 ± 0	0.974 ± 0
F-NIDS	0.885 ± 0.28	0.970 ± 0.264	0.962 ± 0

possibilities, it can be concluded that the DP-SGD algorithm, with the training hyperparameters used, results in negligible penalties on the binary effectiveness of the classifier. Similar behavior was verified when evaluating the effect of the FedAvg algorithm on the classifier by comparing the ANN (Figure 6a) and FED (Figure 6c) methods. In these cases, centralized methods were found to have an advantage over federated methods only in identifying true attacks.

The experimental results are consistent with the established hypotheses, indicating that higher performance metrics could potentially be achieved by reducing the level of Gaussian noise or decreasing the number of DA's used. However, it is important to note that the implementation of these reductions my compromise the system's robustness against membership inference or model inversion attacks. Lower noise levels may weaken the system's resistance to hypothetical scenarios of agent's model theft, while reducing the number of detection agents may make it vulnerable to sample theft, compromising system privacy by inferring the training dataset.

In addition, maintaining a useful classification model for accurate predictions is essential. Therefore, striking a balance between performance and robustness is crucial, with the organization's needs as the primary consideration when using F-NIDS as an intrusion detection tool. For example, in medical applications where privacy has a high importance and predictions are expected to be reviewed by trained professionals, lower levels of accuracy may be acceptable. On the other hand, in general-purpose web applications where accuracy is critical and data privacy may not be a main concern, lower levels of noise may be applicable to achieve higher accuracy. Thus, the system/application designer should carefully evaluate the specific needs and priorities of the organization and the intended application when determining the optimal trade-off between performance and robustness for F-NIDS.

4.4. Robustness against rule-based member inference attacks

To find the best Gaussian noise hyperparameter in the model using the `DP-SGD` algorithm, nine F-NIDS instances containing different values for the σ noise were trained. Each of these detectors were then subjected to rule-based membership inference attacks. This attack approach the scenario where a malicious agent is able to invade a DA, extract the federated classification model, and then use that model to predict training examples that may be stored in some other DA in the network, thus, corrupting the classifier into a predictor of members and non-members of a confidentially accessible dataset.

The criterion chosen to define the best F-NIDS configuration was the instance that held the following indicators: binary accuracy greater than 80% and multi-class accuracy greater than 60%; robustness against real member inference attacks less than 65% and non-member inference attacks less than 40%, yielding an attack efficiency baseline of less than 60%.

In this test the module `MembershipInferenceRuleBase` of the library ART (Adversarial Robustness Toolbox[6]) was used. To produce the results, 20 test cycles were performed. In each of these cycles, the nine models were tested with a set containing a 5% fraction of the original training data examples (members) and 25000 non-member examples.

Figure 7a presents the accuracy results of a membership inference attack, along with the accuracy of the model used. In this figure it is possible to observe three main behaviors. The first is that the higher the value of the Gaussian noise, the lower the obtained attack baseline and the probability of success in inference of non-members. Second, there is an increase in the ability to detect non-members as the value of σ increases, but it does not exceed 50% accuracy. The last notable behavior concerns the model's detection accuracy. The binary and multi-class accuracy decreases, as expected, but it remains significantly above the σ value, and in the case of the binary one, it still maintains very useful prediction levels. Based on the results, the chosen noise value for F-NIDS is $\sigma = 21$, as it can achieve the established criteria of robustness against member inference, yet still be able to maintain intrusion detection accuracy above 80%. Table 3 presents some relevant results from the test performed such attack baseline, attack membership accuracy, binary classification accuracy and the F1 Score obtained by the attack.

Another important indicator is presented in Figure 7b, which contains the precision over the obtained recall on attack's results. Note that the precision of the attack's output isn't significantly affected by increasing the level of noise. However, the recall has presented a decrease as the level of noise is increased. The results show that, while the probability of a non-member being considered a member is not significantly affected by σ, the probability of an actual member being considered a non-member has a rather considerable increase as the level of noise is increased.

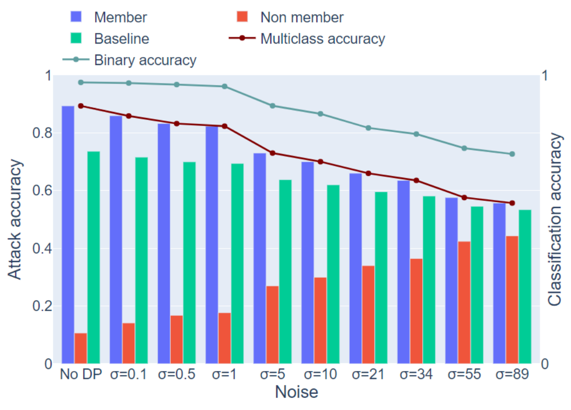
σ	**Atk baseline acc**	**M. Atk acc**	**Bin. acc.**	**Adv. F1 Scr.**
0,1	0.72 ± 0.0010	0.86 ± 0.0010	0.97 ± 0.0010	0.83 ± 0.0007
1	0.69 ± 0.0008	0.82 ± 0.0011	0.96 ± 0.0014	0.81 ± 0.0006
21	0.60 ± 0.0013	0.66 ± 0.0015	0.82 ± 0.0029	0.72 ± 0.0012
55	0.55 ± 0.0016	0.58 ± 0.0016	0.75 ± 0.0030	0.71 ± 0.0012
89	0.53 ± 0.0016	0.56 ± 0.0015	0.73 ± 0.0030	0.66 ± 0.0013

Table 3: Performance of the rule-based adversarial attack

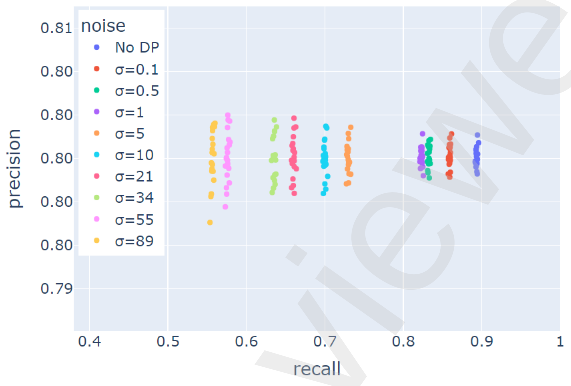
4.5. Robustness against attacks using an adversarial model

Although the classifier used by F-NIDS is protected with differential privacy, if a large amount of data is distributed over a few DA's, it can still represent a risk to be considered.

[6]`https://adversarial-robustness-toolbox.org/`



(a) Robustness and classification performances



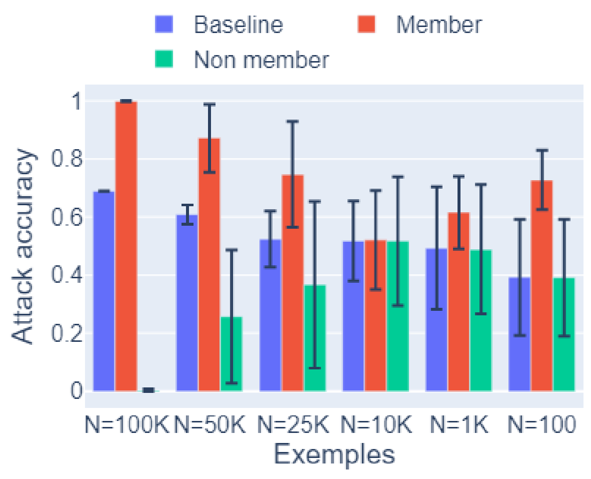
(b) Attack's precision and recall

Figure 7: Attack performances over different F-NIDS classification models. Each result is followed by its ± standard deviation.

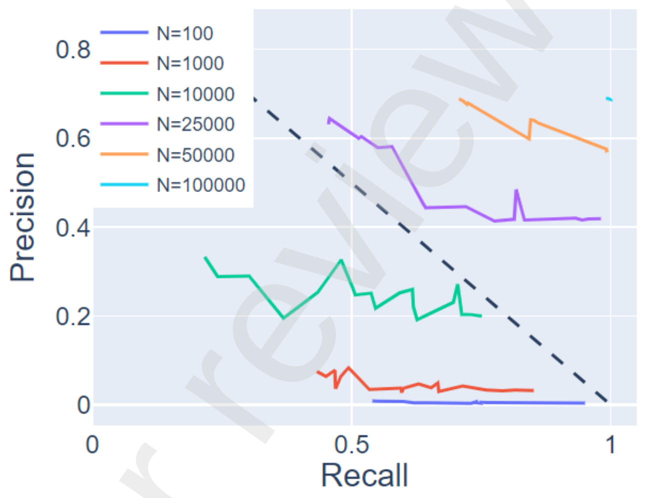
This is because, depending on its size, a sample may contain statistical properties quite identical to those of the original dataset. Thus, such a sample can be used to train a very accurate adversarial model, called *ShadowModel*. This adversarial model can then be used to predict members of confidential datasets from other DA's. Federated learning, used in F-NIDS, plays an important role by allowing to split massive amounts of data to be distribute the training and detection tasks on large quantity of agents, allowing smaller fractions of confidential data to be stored. However, it is still necessary to investigate a maximum number of training examples that can be stored in the DA's, to reduce as much as possible the risk of being used to train an accurate and precise ShadowModel.

For this investigation, six samples from the training dataset, of distinct sizes, were selected and used in an adversarial attack of Membership Inference Black Box, contained in ART library as well. In this attack, six classifiers were trained, each with its respective sample, and the performance of the member and non-member inference accuracy metrics was evaluated, as well as the precision and recall obtained in the attacks. Each of the mentioned samples has a size N of member examples used for training the ShadowModel. This process was repeated in 20 independent cycles and the results is showed in the Table 4. Each cycle was run with $100K$ member examples and $25K$ non-member examples.

The figure 8a presents the effectiveness of the attacks obtained by classifiers trained with different values of N. The vertical axis shows the percentage of attack success and horizontally the obtained results. It can be seen that the effectiveness in detecting members and non-members decreases as the value of N decreases. When this value of $N = 100$ is reached, it coincides with the lowest value recorded for the baseline of the attack accuracy, as both the detection accuracy values for members and non-members remained below 40%. However, a classifier trained with so low examples size would probably will not be of much useful for intrusion detection. Therefore, a number of training examples $10K \leq N \leq 25K$ reaches the reasonably acceptable level for the three indicators present in the figure, where both indicators have values close to 50% attack efficiency. Table 4 presents the obtained indicators, including the baseline, membership inference accuracy and the F1 Score of the attack.



(a) Precision



(b) Recall

Figure 8: Multi-class precision and recall results.

In figure 8b the precision and recall, obtained in the attack results, were compared. Through the graph it is possible to conclude that the precision has a significant increase as the number of training examples is increased. This indicates that the greater the number of examples captured by a malicious agent the greater the chance of building a model capable of inferring members. However, recall did not show great sensitivity to a number of examples used in training the adversarial model, which influences the ability of this model to correctly categorize non-members.

Table 4: Attack's performance indicators using ShadowModels.

N	Baseline	Member	F1 Score adversarial
100000	0.68 ± 0.00	0.99 ± 0.002	0,81 ± 0.001
50000	0.60 ± 0.03	0.87 ± 0.12	0,72 ± 0.06
25000	0.52 ± 0.1	0.74 ± 0.2	0,58 ± 0.11
10000	0.51 ± 0.13	0.52 ± 0.16	0,33 ± 0.07
1000	0.49 ± 0.21	0.61 ± 0.1	0,08 ± 0.03
100	0.39 ± 0.19	0.72 ± 0.1	0,010 ± 0.003

4.6. Robustness against generative attacks using model inversion

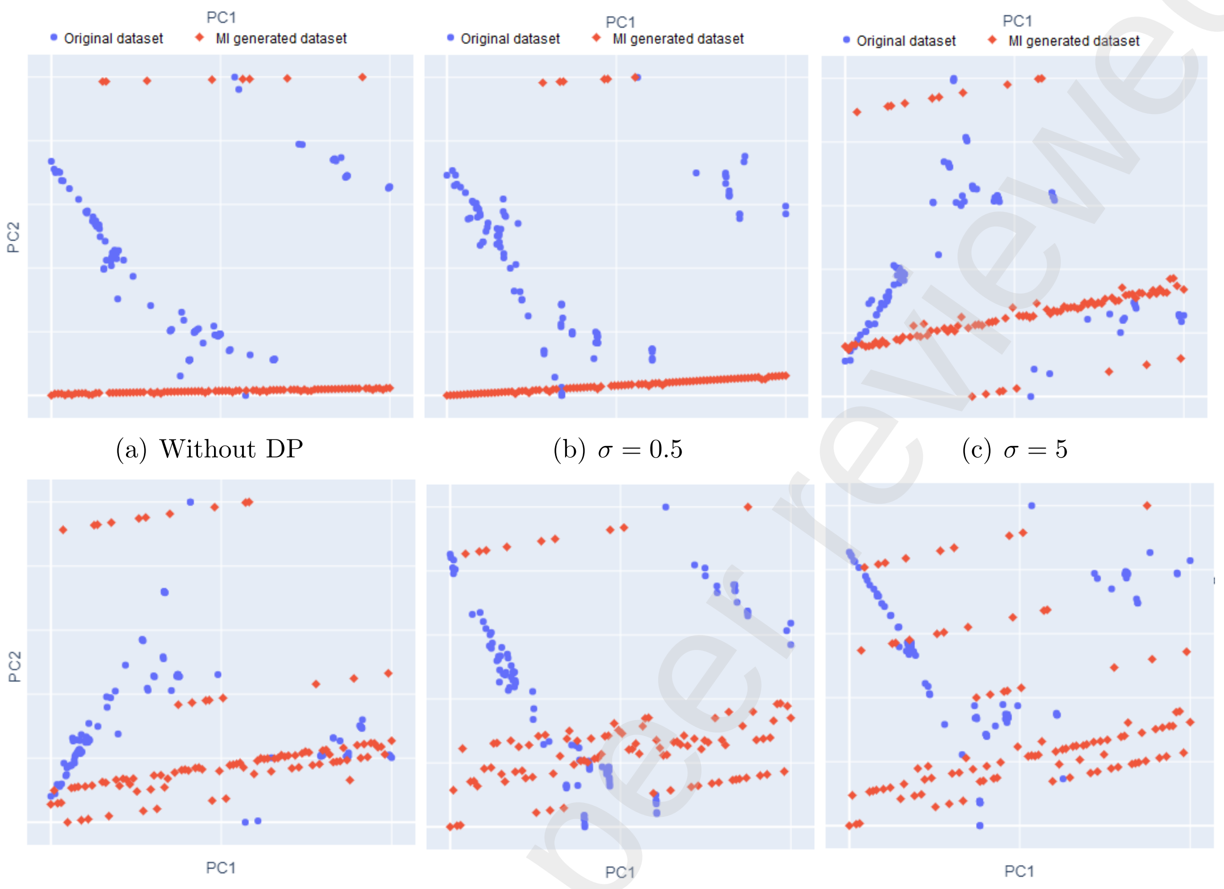
The last robustness test performed on the F-NIDS classifier is the MI (Model Inversion). The MI technique can be used by malicious agents to recreate examples of the original dataset from the trained model. To perform this action, a DA needs to be hacked and its model stolen. The malicious agent then uses the statistical properties that exist in the gradients to generate a mass of data that may contain original examples of other DA's, thus violating their confidentiality. Through this test it was possible to obtain a major difference in the generated data as the level of noise in the original classification model is increased. The MI algorithm used was `MIFace` [37], available in the ART library.

To validate the test, 100 examples were extracted from models trained with different levels of noise. The models used in this test are the same ones used in the membership inference test, already discussed in the 4.4 sessions. For each of the generated datasets, an original set containing the same amount of examples and a similar class distribution was randomly selected. Both of these datasets had their dimensionality reduced to two principal components, using PCA (Principal Component Analysis) technique, aiming to obtain a graphical representation of the effect of the Gaussian noise on the datasets.

Figure 9 graphically presents the results of the original and adversarial datasets, generated by the six models trained using different noise levels, with their dimensionality reduced to two principal components. It can be seen in Figure 9a with no DP algorithm used on the target model, the MI generated dataset shows a relatively similar pattern to the original, where the points are located in two well-defined sets, with similar distribution. Other important aspect is that it is possible to draw one same decision boundary to distinguish two regions for both datasets. As the level of noise increases, it can be seen that the generated examples show an increasingly distinct pattern from the original dataset. Thus, the recreated examples begin to show a very distinct dispersion pattern from the original, especially when $\sigma = 89$, according to Figure 9f. Leading to the conclusion that in a dataset produced from a ShadowModel, with a higher level of noise, it becomes, for example, more complex for some classification algorithm to draw a decision boundary who fits both datasets. Another issue to note is that, with $\sigma \geq 5$ present in Figure 9c, it is no longer possible to draw a same decision region that can linearly separate both training and adversarial datasets into two decision boundaries with similar distribution. It leads to conclude that both training and adversarial datasets become to much distinguished each other when Gaussian noise is increased to five or above on the target model. Therefore, the adversarial data becomes inappropriate for training an adversarial model that is capable of inferring properly unknown samples from the original dataset.

5. Conclusion and Future Research

This research showed that one of the main challenges for IoT is decentralized and scalable intrusion detection in these networks. Despite attempts to achieve accurate detections in an IoT environment, there was still directions for improvement in terms of scalability, confidentiality, and reliability. The results showed that, with lower Gaussian noise values, F-NIDS has similar accuracy, precision, and recall metrics as traditional strategies that use only centralized learning. Considering the results obtained, F-NIDS showed to be effective to: (i) efficiently detect eventual attacks and satisfactorily determine which attack is being carried out, among a list of the most common attacks; (ii) train models in a distributed manner while maintaining the privacy of the training data; and (iii) be endowed with a decentralized classification architecture that allows its use in IoT networks of the most diverse



(a) Without DP (b) $\sigma = 0.5$ (c) $\sigma = 5$

(d) $\sigma = 21$ (e) $\sigma = 55$ (f) $\sigma = 89$

Figure 9: Reduced dimensions to two components of the original and adversarial datasets.

applications, ensuring the scalability and reliability of the system.

Furthermore, the results allow to define good settings to two hyperparameters which are very important to guarantee confidentiality, while keeps the classification performance. This hyperparameters are the Gaussian noise σ and detection agent's training sample size N. The tests showed that $\sigma = 21$ protects against membership inference black-box rule based attacks and model inversion attacks. Keeping sample size, stored on detection agents, between $N = 10K$ and $N = 25K$ helps to protect the detection agents confidentiality, in case of conventional membership inference black-box attacks, for instance, if some agent has their confidentiality compromised, other detection agents can keep his own training data privacy ensured.

Although confidentiality is a very important pillar of information security, data integrity and availability also are equally important factor to be recon with. Therefore F-NIDS needs to have metrics that approach these other pillars too. Keeping this in mind, in future works, other adversarial machine learning testings can be performed on classification models, to measure his robustness against other adversarial ML attack types. The other adversarial threats that can compromise the system classification model are the evasion and poisoning.

These attacks are related the other mentioned security pillars and performing measurements, thus, they can help to improve the F-NIDS robustness not only about confidentiality, but also data integrity and availability.

Acknowledgment

This research was partially funded by the FAPESP (São Paulo Research Foundation) grant #2020/05126-6, FAPEMIG (Minas Gerais Research Foundation) grant #APQ-02675-21.

References

[1] M. A. Rahman, A. T. Asyhari, The emergence of internet of things (iot): Connecting anything, anywhere, Computers 8 (2). doi:10.3390/computers8020040.
URL `https://www.mdpi.com/2073-431X/8/2/40`

[2] P. Geraldo Filho, L. A. Villas, V. P. Gonçalves, G. Pessin, A. A. Loureiro, J. Ueyama, Energy-efficient smart home systems: Infrastructure and decision-making process, Internet of Things 5 (2019) 153–167.

[3] H. Habibzadeh, B. H. Nussbaum, F. Anjomshoa, B. Kantarci, T. Soyata, A survey on cybersecurity, data privacy, and policy issues in cyber-physical system deployments in smart cities, Sustainable Cities and Society 50 (2019) 101660. doi:https://doi.org/10.1016/j.scs.2019.101660.
URL `https://www.sciencedirect.com/science/article/pii/S2210670718316883`

[4] R. Roman, J. Zhou, J. Lopez, On the features and challenges of security and privacy in distributed internet of things, Computer Networks 57 (10) (2013) 2266–2279, towards a Science of Cyber Security Security and Identity Architecture for the Future Internet. doi:https://doi.org/10.1016/j.comnet.2012.12.018.
URL `https://www.sciencedirect.com/science/article/pii/S1389128613000054`

[5] I. C. Cavalcante, R. I. Meneguette, R. H. Torres, L. Y. Mano, V. P. Gonçalves, J. Ueyama, G. Pessin, G. D. Amvame Nze, G. P. Rocha Filho, Federated system for transport mode detection, Energies 15 (23) (2022) 9256.

[6] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, P. Faruki, Network intrusion detection for iot security based on learning techniques, IEEE Communications Surveys Tutorials 21 (3) (2019) 2671–2701. doi:10.1109/COMST.2019.2896380.

[7] E. Bertino, N. Islam, Botnets and internet of things security, Computer 50 (2) (2017) 76–79. doi:10.1109/MC.2017.62.

[8] M. A. Rahman, A. T. Asyhari, L. Leong, G. Satrya, M. Hai Tao, M. Zolkipli, Scalable machine learning-based intrusion detection system for iot-enabled smart cities, Sustainable Cities and Society 61 (2020) 102324. doi:https://doi.org/10.1016/j.scs.2020.102324.
URL `https://www.sciencedirect.com/science/article/pii/S221067072030545X`

[9] J. Cabrero-Holgueras, S. Pastrana, Sok: Privacy-preserving computation techniques for deep learning, Proceedings on Privacy Enhancing Technologies 2021 (4) (2021) 139–162. doi:doi:10.2478/popets-2021-0064.
URL `https://doi.org/10.2478/popets-2021-0064`

[10] R. S. Siva Kumar, M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, S. Xia, Adversarial machine learning-industry perspectives, in: 2020 IEEE Security and Privacy Workshops (SPW), 2020, pp. 69–75. doi:10.1109/SPW50608.2020.00028.

[11] H. Chen, S. U. Hussain, F. Boemer, E. Stapf, A. R. Sadeghi, F. Koushanfar, R. Cammarota, Developing privacy-preserving ai systems: The lessons learned, in: 2020 57th ACM/IEEE Design Automation Conference (DAC), 2020, pp. 1–4. doi:10.1109/DAC18072.2020.9218662.

[12] H. Zhu, H. Zhang, Y. Jin, From federated learning to federated neural architecture search: a survey, Complex & Intelligent Systems 7. doi:10.1007/s40747-020-00247-z.

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y. Arcas, Communication-Efficient Learning of Deep Networks from Decentralized Data, in: A. Singh, J. Zhu (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Vol. 54 of Proceedings of Machine Learning Research, PMLR, 2017, pp. 1273–1282.
URL `https://proceedings.mlr.press/v54/mcmahan17a.html`

[14] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 3–18. doi:10.1109/SP.2017.41.

[15] C. Dwork, Differential privacy, in: M. Bugliesi, B. Preneel, V. Sassone, I. Wegener (Eds.), Automata, Languages and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 1–12.

[16] W. L. Al-Yaseen, Z. A. Othman, M. Z. A. Nazri, Real-time multi-agent system for an adaptive intrusion detection system, Pattern Recognition Letters 85 (2017) 56–64. doi:https://doi.org/10.1016/j.patrec.2016.11.018.
URL `https://www.sciencedirect.com/science/article/pii/S0167865516303415`

[17] R. M., M. Ahmed, R. Khan, An adaptive distributed intrusion detection system architecture using multi agents, International Journal of Electrical and Computer Engineering (IJECE) 9 (2019) 4951. doi:10.11591/ijece.v9i6.pp4951-4960.

[18] G. Raja, S. Anbalagan, G. Vijayaraghavan, S. Theerthagiri, S. V. Suryanarayan, X.-W. Wu, Sp-cids: Secure and private collaborative ids for vanets, IEEE Transactions on Intelligent Transportation Systems 22 (7) (2021) 4385–4393. doi:10.1109/TITS.2020.3036071.

[19] A. Tabassum, A. Erbad, A. Mohamed, M. Guizani, Privacy-preserving distributed ids using incremental learning for iot health systems, IEEE Access 9 (2021) 14271–14283. doi:10.1109/ACCESS.2021.3051530.

[20] R. Zhao, Y. Yin, Y. Shi, Z. Xue, Intelligent intrusion detection based on federated learning aided long short-term memory, Physical Communication 42 (2020) 101157. doi:https://doi.org/10.1016/j.phycom.2020.101157.
URL `https://www.sciencedirect.com/science/article/pii/S1874490720302342`

[21] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, E. Ilie-Zudor, Chained anomaly detection models for federated learning: An intrusion detection case study, Applied Sciences 8 (12). doi:10.3390/app8122663.
URL `https://www.mdpi.com/2076-3417/8/12/2663`

[22] E. ul Haq Qazi, M. Imran, N. Haider, M. Shoaib, I. Razzak, An intelligent and efficient network intrusion detection system using deep learning, Computers and Electrical Engineering 99 (2022) 107764. doi:https://doi.org/10.1016/j.compeleceng.2022.107764.
URL `https://www.sciencedirect.com/science/article/pii/S0045790622000684`

[23] U. Mbasuva, G.-A. L. Zodi, Designing ensemble deep learning intrusion detection system for ddos attacks in software defined networks, in: 2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM), 2022, pp. 1–8. doi:10.1109/IMCOM53663.2022.9721785.

[24] J. Yu, X. Ye, H. Li, A high precision intrusion detection system for network security communication based on multi-scale convolutional neural network, Future Generation Computer Systems 129 (2022) 399–406. doi:https://doi.org/10.1016/j.future.2021.10.018.
URL `https://www.sciencedirect.com/science/article/pii/S0167739X21004143`

[25] A. Salem, Y. Sautter, M. Backes, M. Humbert, Y. Zhang, Baaan: Backdoor attacks against autoencoder and gan-based machine learning models.

[26] P. Ruzafa-Alcazar, P. Fernandez-Saura, E. Marmol-Campos, A. Gonzalez-Vidal, J. L. Hernandez Ramos, J. Bernal, A. F. Skarmeta, Intrusion detection based on privacy-preserving federated learning for the industrial iot, IEEE Transactions on Industrial Informatics (2021) 1–1doi:10.1109/TII.2021.3126728.

[27] A. Kundu, Fed+: A unified approach to robust personalized federated learningdoi:10.13140/RG.2.2.27907.43040.

[28] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawit, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. El Rouayheb, D. Evans, J. Gard-

ner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. Theertha Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, S. Zhao, Advances and Open Problems in Federated Learning, Now Foundations and Trends, 2021.

[29] N. Sai Lohitha, M. Pounambal, Integrated publish/subscribe and push-pull method for cloud based iot framework for real time data processing, Measurement: Sensors 27 (2023) 100699. doi:https://doi.org/10.1016/j.measen.2023.100699.
URL `https://www.sciencedirect.com/science/article/pii/S2665917423000351`

[30] S. Han, H. Woo, Ndn-based pub/sub system for scalable iot cloud, in: 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2016, pp. 488–491. doi:10.1109/CloudCom.2016.0085.

[31] A. Lazidis, K. Tsakos, E. G. Petrakis, Publish–subscribe approaches for the iot and the cloud: Functional and performance evaluation of open-source systems, Internet of Things 19 (2022) 100538. doi:https://doi.org/10.1016/j.iot.2022.100538.
URL `https://www.sciencedirect.com/science/article/pii/S2542660522000403`

[32] Z. Bu, J. Dong, Q. Long, W. Su, Deep Learning With Gaussian Differential Privacy, Harvard Data Science Review 2 (3), https://hdsr.mitpress.mit.edu/pub/u24wj42y.

[33] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy (2016) 308–318doi:10.1145/2976749.2978318.
URL `https://doi.org/10.1145/2976749.2978318`

[34] P. Kácha, Idea: security event taxonomy mapping, in: 18th International Conference on Circuits, Systems, Communications and Computers, 2014.

[35] H. Debar, J. Viinikka, Intrusion detection: Introduction to intrusion detection and security information management, Vol. 3655, 2005, pp. 207–236. doi:10.1007/11554578_7.

[36] M. Sarhan, S. Layeghy, M. Portmann, Towards a standard feature set for network intrusion detection system datasets, Mobile Networks and Applications 27 (1) (2022) 357–370. doi:10.1007/s11036-021-01843-0.
URL `https://doi.org/10.1007/s11036-021-01843-0`

[37] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 1322–1333. doi:10.1145/2810103.2813677.
URL `https://doi.org/10.1145/2810103.2813677`