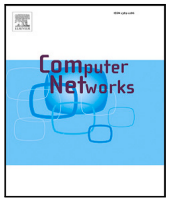
Computer Networks 236 (2023) 110010

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



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], José C.M. Oliveira [d], Geraldo P. Rocha Filho [d]

[a] University of Brasília, Department of Electrical Engineering, Brasília - 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 Exact and Technological Sciences, State University of Southwest Bahia, Candeias, BA, Brazil

ARTICLE INFO

Keywords:
Network intrusion detection system
Federated learning
Asynchronous messaging
Security systems
Distributed computing
Pub/sub mechanism

ABSTRACT

The rise of IoT networks has presented fresh challenges in terms of scalability and security for distributed Network Intrusion Detection Systems (NIDS) due to privacy concerns. While some progress has been made in addressing these challenges, there are still unanswered questions regarding how to achieve a balance between performance and robustness to ensure privacy in a distributed manner. Additionally, there is a need to develop a reliable and scalable architecture for distributed NIDS that can be effectively deployed in various IoT scenarios. These questions about robustness relied mainly on choosing privacy-secured and distributed Machine Learning techniques. In this work, we propose the F-NIDS, an intrusion detector that utilizes federated artificial intelligence and asynchronous communication techniques between system entities to provide horizontal scalability, along with differential privacy techniques to address data confidentiality concerns. The architecture of F-NIDS is designed to be adaptable for usage in IoT networks, suited to be used in cloud or fog-based environments. Results from our experiments have shown that the confidential detection model employed in F-NIDS – considering multi-class accuracy, binary accuracy, precision, and recall metrics – was 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.

1. Introduction

In 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 [3,4]. Decentralized architectures may bring some solutions to these issues, offering higher availability and superior scalability [5,6]. 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 [7]. By the other hand, conventional NIDS techniques may be less effective in the IoT context, due to their dynamism [8].

A NIDS is built to provide continuous monitoring and detection during the life cycle of computer networks [9]. 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 [8]. Therefore, NIDS works under more challenging and restrictive circumstances when used in this environment. On

* Corresponding author.
E-mail addresses: jonathas.oliveira@aluno.unb.br (J.A. de Oliveira), vpgvinicius@unb.br (V.P. Gonçalves), meneguette@icmc.usp.br (R.I. Meneguette), desousa@unb.br (R.T. de Sousa Jr), guidoni@ufop.edu.br (D.L. Guidoni), jcarlos@uesb.edu.br (J.C.M. Oliveira), geraldo.rocha@uesb.edu.br (G.P. Rocha Filho).

https://doi.org/10.1016/j.comnet.2023.110010
Received 1 June 2023; Received in revised form 7 August 2023; Accepted 30 August 2023
Available online 9 September 2023

the occasion of the excellent results, researchers have adopted the ML approach (Machine Learning) for NIDS development to improve cyberattack detection [7]. On the other hand, the authors discussed in [10] the issue of privacy in ML models still remains a challenge, especially in 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 [11,12]. 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 [13,14]. Despite FL being used for distributing ML training tasks, another important question is about distributing the detection services across multiple agents over the network. The Asynchronous communication approach comes to give such capability to the systems.

The asynchronous communication approach is based on message queuing, callbacks, event-driven architecture, and publish/subscribe model techniques. In [15] a relevant overview of messaging patterns and architectures is provided, including asynchronous communication, message queuing, and event-driven systems. Additionally, [15] offers a collection of patterns for designing asynchronous messaging systems, covering concepts such as message queues, publish/subscribe models, and event-driven architectures. The benefits of asynchronous communication in distributed systems are discussed in [16], highlighting the advantages of decoupled, event-driven architectures. However, this approach has some limitations and trade-offs, such as the complexity of implementing and handling errors. The latency times also can be increased [17].

The goal of this work is to propose the F-NIDS, a distributed intrusion detection system in order to overcome the mentioned privacy limitations in such distributed scenarios by evaluating the classification performance and robustness in terms of accuracy, precision, and recall metrics. A distributed architecture is presented to provide distributed intrusion detection and horizontal scalability over distributed IoT scenarios. 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 2 presents the related articles. Section 3 describes the F-NIDS. Section 4 presents the results through performance evaluation. Section 5 concludes the paper with the conclusion and future work.

2. Federated learning

Federated Learning (FL) is a machine learning technique that allows collaborative learning on distributed data while maintaining some levels of data privacy. Introduced in 2016, FL enables multiple devices to collectively train a shared model by performing local training on their data without sharing it centrally [18]. This is possible by exchanging a model weight between the training device and the model aggregator, which transforms multiple models into one global model. This decentralized training paradigm mitigates privacy concerns and communication overhead [19].

The process involves initializing a global model on a central server, selecting devices to participate, performing local training on each device's data, aggregating the updated models on the central server, and iterating these steps until convergence or desired model performance is achieved [18–21]. By keeping data local and avoiding data sharing, federated learning addresses privacy concerns while reducing communication costs [19], making it suitable for scenarios with distributed data sources and many devices [22].

FL offers a scalable approach for distributing model training tasks. In this decentralized learning paradigm, the training process occurs on local devices or agents, allowing for parallelization and efficient training. This distributed model training process reduces communication overhead and enhances scalability. The scalability of federated learning has been highlighted in various studies, including [23], demonstrating its ability to handle large-scale distributed learning tasks. However, [23] states that one of the main challenges is striking the right balance between privacy preservation and model accuracy, as introducing noise for privacy protection can impact the utility of the trained model. Besides effectively managing the privacy budget, quantifying the amount of privacy protection provided is another challenge in federated learning with differential privacy. Overhead, synchronization, and resource management to accommodate training with a large number of federated members is another important challenge to be addressed [19,23].

FL also enables integration with DP (Differential Privacy) mechanisms, mitigating some of the limitations of NIDS. While keeping isolated training data on clients may, in theory, increase confidentiality, the risk remains from the inference of information from the trained models [24]. DP ensures that any version of a statistical dataset remains equally credible regardless of whether it contains a particular item [25], minimizing the possibility of inference of sensitive individual information but maintaining the statistical properties of the dataset.

3. Related works

Centralized NIDS solutions have been recently developed aiming to extend the level of accuracy and performance of intrusion detections [26–28]. Specifically, in [29] they propose the use of convolutional neural networks (ConvNet) in intrusion detection systems. In [30], a deep neural network is defined with the stacking of asymmetric autoencoders, 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 [7].

In the context of distributed NIDS's, different solutions have been proposed [31,32]. 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 [7].

Other decentralized NIDS solutions, which explore the issue of data privacy and reliability with more emphasis, have been investigated in [33,34] proposes SP-CIDS, an ML solution for distributed NIDS aimed at serving autonomous vehicle networks, by applying DP techniques to the training data. [34] 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 [33] 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

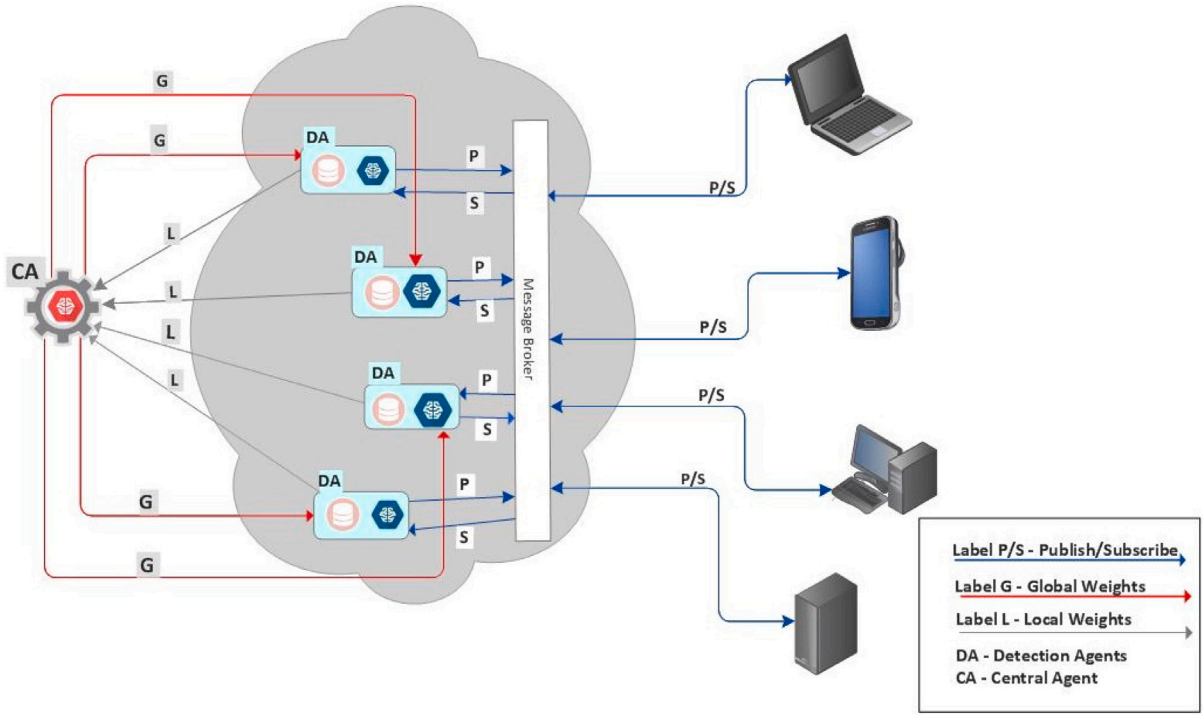


Fig. 1. General F-NIDS architecture.

the transfer of raw data between a central agent, and are therefore ineffective for privacy protection [33,35]. Furthermore, both works use the assumption that generative neural networks inherently produce more confidential models for storage and transmission, however as demonstrated in [36], such ML techniques can also be vulnerable to confidentiality violations, specifically inference attacks on the training data.

In [37], 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 [38]. It is worth noting that [37] 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 [23].

Some works have already been made proposing systems that use pub/sub mechanisms to provide communication between system entities. Thus, validating the scalability and availability provided by such a technique. In [39] 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. On the other hand, [40] proposed a communication solution that applies pub/sub technique in conjunction with NDN (Named Data Networking) and leverages the availability and scalability of the data exchange between entities. The evaluations performed in [41] 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 a cluster. Thus, maintaining fault tolerance, and supporting heavy workloads, have proven to be scalable solutions.

Table 1 provides a comparison between some of the other recent IDS works in terms of advantages and limitations. In this table, the differences between recent NIDS works are highlighted, such as the kind, in terms of architecture and used techniques; number of agents used in system training and performance evaluation; the age of the work; the system performance in terms of accuracy, and also advantages and limitations of each studied IDS. Its possible to notice that, different of other works, F-NIDS was tested in with a significant number of agents, and will able to provide balanced model with robustness indicators, with relatively low performance cost.

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 scalability 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 a higher number of detection agents and transmitting just the trained weights, using federated learning. In the 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.

4. F-NIDS — federated network intrusion detection system

This section presents the F-NIDS, a federated intrusion detection system coupled with DP, to allow detection service horizontal scalability. The system uses FL to distribute the training tasks, orchestrate

Table 1
Comparative study.

System	Authors	Year	Kind	Agents	Performance	Advantages	Limitations
FELIDS	Friha et al.	2022	Distributed FL non-DP	15	90.05% accuracy	Many classifiers evaluation	Lack of privacy preserving mechanism; low amount of clients
Fed+ IDS	Ruzafa-Alcáza et al.	2021	Distributed FL + DP	5	89% accuracy	Many DP techniques evaluation	Low amount of clients; no robustness evaluation
MS CNN	Jing et al.	2022	Centralized CNN	1	95.7% accuracy	High performance metrics	Single point of failure
F-NIDS	Oliveira et al.	2023	Distributed FL + DP	100	88% accuracy	Robustness analysis; Balanced accuracy and robustness; Detection architecture design	

the data exchange between entities via federated rounds, and aggregate the local models via FedAvg (Federated Averaging) algorithm. In addition, the DP technique is used by F-NIDS to deliver an additional security layer by protecting the confidentiality of the model's data. This technique applies the DP-SGD (Differentially Private-Stochastic Gradient Descent) algorithm to train local models by applying noise on the model's local weights during the gradient descent algorithm execution at the agent's side. Then the local models can be stored and exchanged between agents in a secure way. The distributed detection service is implemented via publish/subscribe mechanism.

4.1. Overview

The F-NIDS architecture is illustrated in Fig. 1. In the figure, the central agent (label CA) generates the initial global weights (label G), propagating them to the other system members. The CA has the function of aggregating weights and producing a global model using a local weight aggregation technique and also performs the orchestration, propagating the weights obtained in this aggregation. The detection tasks, reception of the global weights, and training of a local model are done by the Detection Agents (label DA). Newer DAs can be added to this system, scaling the training tasks. Besides training a local model using an individual dataset, this mechanism allows splitting small training tasks between heterogeneous DAs with lower network bandwidth consumption making it suitable for IoT-distributed scenarios.

The MB handles asynchronous tasks using the pub/sub model and managing queues. It queues clients' detection message requests and the DA's responses into the appropriate queues. In addition, both clients and detection agents have a pub/sub driver to listen for events, publish messages, and subscribe to the MB. Communication between these agents occurs through MQTT (Message Queuing Telemetry Transport), a lightweight protocol enabling the handling of asynchronous events and the decoupling of agents. Three types of queues are utilized: the detection request queue, the notification queue (one for each client), and the alerts queue. The detection request queue receives messages from clients that are subscribed to by all detection agents, while notification queues ensure specific clients receive relevant notifications published by the detection agents. The alerts queue, subscribed to by all clients, allows for the publication of alerts by the DAs. It will allow the system to scale horizontally, be fault-tolerant, and be more available.

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 GA 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 important aspect important to emphasize 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 DAs, 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.

Table 2
List of hyperparameters used in model training.

Hyperparameter	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

F-NIDS was designed to be deployed in the cloud, with the clients performing intrusion detection requests through the internet. In a cloud deployment, all the DAs and the MB will be located geographically apart from the clients. However, it can be also deployed in a fog, as well. In that case, the system architecture supports detection agents being deployed locally, closer to the clients. Just the central agent is kept over the internet, just for the model's aggregation purposes. As the model has security measures to prevent malicious exploitation, due to the DP mechanism, such an approach stays secure to be used. The distributed characteristics allow fast horizontal scaling, which makes it suited for IoT scenarios.

4.2. 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 2) 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 via `FedAvg` algorithm. The federated round concludes when the propagation of the federated model parameters to the detection agents is finished.

In the `FedAvg` [18] 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 the CA. In this strategy, the computational cost is controlled by three hyperparameters: C, 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 the DAs registered in F-NIDS, passing as an argument the weights of the global model and updating the vector of local weights with the new weight values returned from the DA (line 6). 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.

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

4.3. Classifier's differential privacy mechanism

To ensure the highest confidentiality of the trained models, F-NIDS implements the algorithm DP-SGD. This technique is implemented locally in each DA during the gradient descent algorithm execution. Before the weights are updated, the gradients are clipped, and Gaussian noise is added to it, producing a secure model to be stored or his weights exchanged over the network by the FL algorithm. When AC receives this secure model, the FedAvg algorithm can perform the aggregation tasks normally, but the global model will inherit the privacy properties of local models. This approach is the classic SGD model optimization algorithm version but includes DP. This algorithm limits the sensitivity of each gradient [42]. 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 $\delta \sim \mathcal{N}(0, \sigma^2 C^2 I)$ is the random variable corresponding to Gaussian DP and σ^2 the noise's standard deviation [43].

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 η_t, noise σ, minibatch size L, gradient 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: $\tilde{g}_t \leftarrow \frac{1}{L} \sum_i (\bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 I))$ ▷ //Gradient noise step
10: $w_{t+1} \leftarrow w_t - \eta_t \tilde{g}_t$ ▷ //Adjust weights
11: **end for**
12: **Return** w_T.

4.4. 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. To meet the cases that can be impracticable for the client's capture and send packages to be detected, one kind of special agent can be deployed in the system to capture packages via network probing and send them to the detection agents. In this case, this special agent will act like an IPS (Intrusion Prevention System). 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 Fig. 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 to 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.

Fig. 3 illustrates the communication model between all available DAs and the clients interested in detection intrusion. For the detection of an intrusion, a client c publishes a message m, formatted from a

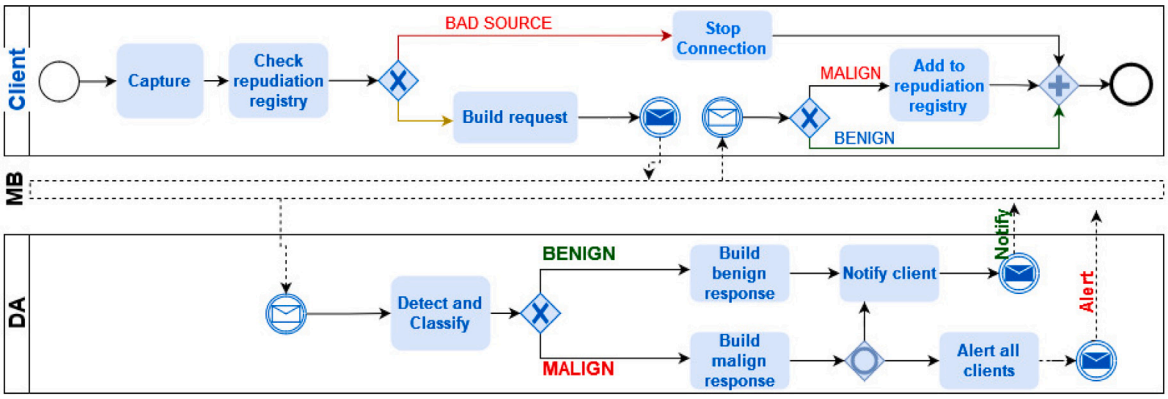


Fig. 2. Basic operation between a certain DA and a client.

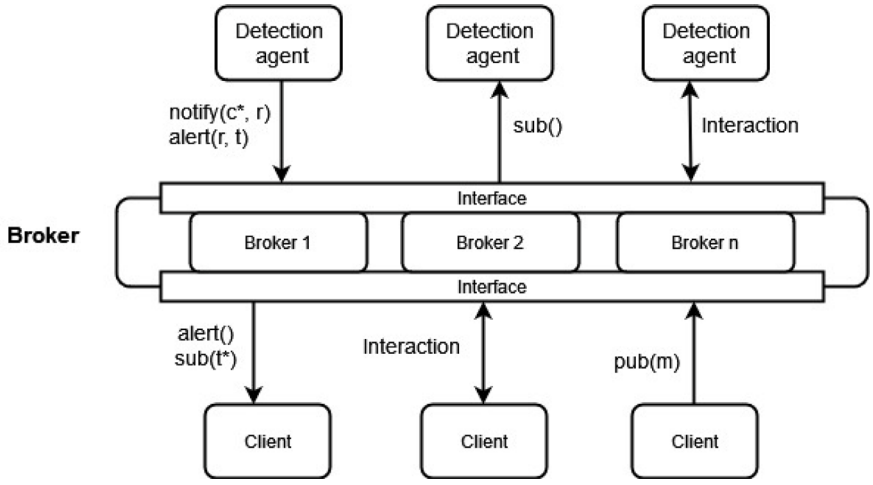


Fig. 3. F-NIDS communication model.

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 a 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 [44], a communication format that uses JSON (Javascript Object Notation) notation and which is based on the IDMEF (Intrusion Detection Message Exchange Format) proposed by [45]. In this standard, 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, which 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.

The main challenge of this system is to strike a balance between privacy and robustness. FL and DP algorithms come with a classification performance cost, and establishing the right balance to keep the helpful model and confidential to be distributed across hundreds of clients is a very challenging task. However, other kinds of challenges are worth mentioning. The first concerns keeping a distributed system easy to deploy and maintain. Besides, it is necessary to take into consideration that can be too much complex the task of installing some software layer on the clients to handle the package capture and sending that to the DAs via the MQTT protocol can be complex, especially for large networks. Besides, debugging a distributed environment can impose a considerable challenge due to the number of agents communicating with each other.

5. 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 prediction performances in terms of accuracy, precision and recall. Each of these F-NIDS performance indicators was compared with the other three methods commonly used in other NIDS works. The accuracy metric aims to verify which round the F-NIDS convergence occurs in comparison with other methods.

The performance of the methods was evaluated using in addition to accuracy, precision, and recall. Precision (P_c) is described in Eq. (1)

and is the measure of the proportion of true positive predictions (TP_c) out of all predictions made for 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 proportion of true positive predictions out of all actual positive instances for a given class described in Eq. (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.

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

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

To compare F-NIDS with other existing ML methods, already studied in other works, the classifier training methods evaluated are 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, each evaluated on a random fraction of 10% of the test dataset. To obtain the overall multi-class and binary performance across the four methods, the average accuracy will be analyzed over the training rounds in Section 5.1. This strategy aims to observe at which round each method reaches relative levels of convergence. The Section 5.2 deals with the evaluation of the obtained accuracy and *recall* results, aiming to verify the classification performance in each of the classes individually, the classes are listed in 5.2. To the evaluations performed on Section 5.4, 5.5, and 5.6, nine instances of the F-NIDS were trained, and each of these instances was submitted to ten different adversarial attacks. The average attack's accuracy, precision, and recall were used to compare each of the results get from each model.

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] [46], [where the produced database has 43 relevant features. For this work, `IPV4_SRC_ADDR and IPV4_DST_ADDR]` – representing the sender and receiver IP addresses information – has been removed. The research used just 41 of the remaining features. 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. The table presents the list of hyperparameters used in the training model. The hyperparameters of the multilayer perceptron were obtained using the *hyperparameter tuning* method, and the size of the minibatch was the largest supported by the GPU used (NVIDIA V100 with 16 GB of GDDR5). The hyperparameters related to federated learning were chosen taking into account the available CPU and RAM memory (8 cores and 24 GB RAM). Other values of hyperparameters were tested but with a lower classifier performance in terms of accuracy and convergence.

5.1. F-NIDS accuracy results

Fig. 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 Fig. 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 has 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.

Fig. 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 and 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.

5.2. F-NIDS performance per class evaluation

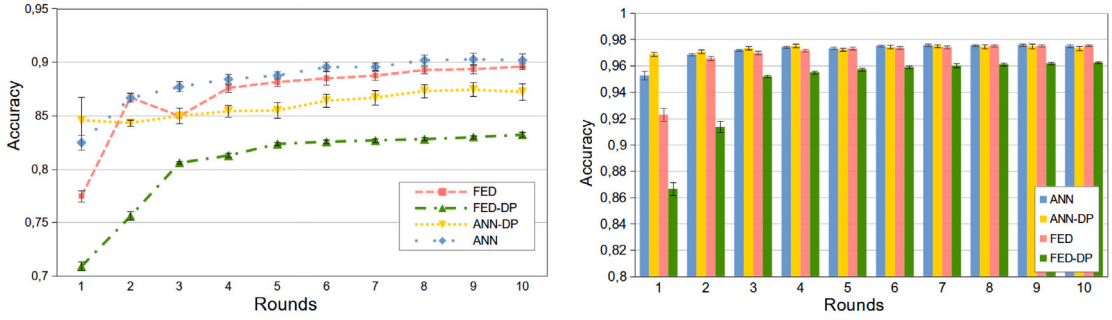
In Fig. 5, the accuracy and *recall* obtained on each class using the F-NIDS classification method is presented and compared to the ANN, ANN-DP, and FED methods. Fig. 5a presents the accuracy, or proportion of hits among the predictions made, demonstrating the ability of the model to correctly classify a class. Fig. 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 number 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 impact on overall accuracy and multi-class recall.

5.3. 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).

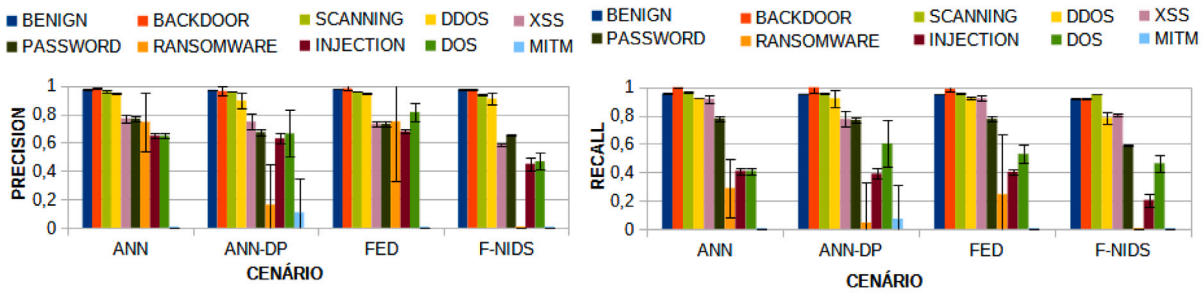
Fig. 6 presents the confusion matrices for each of the methods, showing the intersection between the examples classified as benign or attack and their corresponding actual values. The F-NIDS method (Fig. 6d) obtained a similar result to the others in the correct detection of benign traffic, which also contributed to maintaining 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,

[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/.



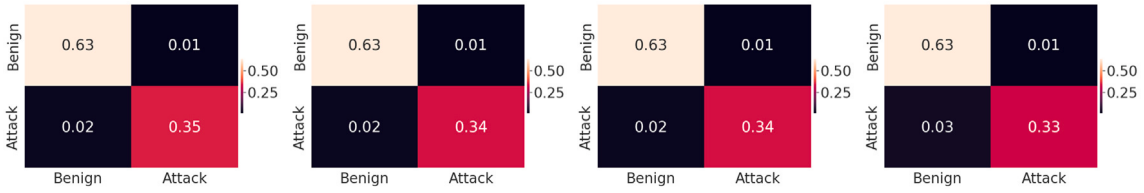
(a) Multiclass accuracy (b) Binary accuracy

Fig. 4. Binary and multiclass accuracy per round.



(a) Precision (b) Recall

Fig. 5. Multi-class precision and recall results.



(a) ANN (b) ANN-DP (c) FED (d) F-NIDS

Fig. 6. Binary confusion matrices of the classifiers.

Table 3
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

represented by Fig. 6a, b and c. 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 (Fig. 6a), ANN-DP (Fig. 6b) and FED (Fig. 6c) methods to detect benign packets and almost as efficient to correctly detect attacks as the other mentioned approaches. Table 3 numerically presents the results of the obtained metrics. It contains the averages of the ten observations and the errors represented by their standard deviation.

When comparing the effects of the DP-SGD algorithm on attack classification performance, by comparing the ANN method (Fig. 6a) with ANN-DP (Fig. 6b), it can be seen that this algorithm slightly influences the ability to detect true attacks. Considering all possibilities, it can be concluded that the DP-SGD algorithm, with the training hyperparameters, 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 (Fig. 6a) and FED (Fig. 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 DAs used. However, it is important to note that the implementation of these reductions may 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 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 high importance and predictions are expected to be reviewed by trained professionals, lower levels of accuracy may be acceptable. On the other hand, in non privacy-critical applications, where high accuracy rate maybe not crucial, such facial recognition for instance, data privacy may not be a main concern. In this case, 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. In the following sections, a vulnerability assessment was conducted with real adversarial attacks to verify which F-NIDS setting can be more robust against confidentiality threads.

5.4. Robustness against rule-based membership inference attacks

Rule-based member inference attacks involve inferring an individual's membership in a sensitive dataset using external knowledge or rules. Adversaries use background information to create rules that predict membership [47]. By applying these rules to an individual's attributes, attackers determine membership, even without direct access to the dataset [48]. Such attacks pose privacy risks when protecting dataset membership [24]. Since the FL technique involves spreading models weights across multiple locations or devices, each specific model itself can be a source of knowledge to be used by one hypothetical malicious agent.

Training the model using the `DP-SGD` algorithm using a Gaussian noise hyperparameter can make the system robust against such attacks. However, it comes with classification performance costs. Therefore, balancing robustness and performance is mandatory to verify the model behavior. Keeping this in mind, nine F-NIDS instances groups containing different values for the σ noise were trained, aiming to select which performs better in the attack, producing as useful classification performance as possible. Each of these detectors was then subjected to rule-based membership inference attacks. The group that presents better robustness against the attack at the lower classification performance cost will have the related noise selected. However, it is necessary to establish a criterion based on performance and robustness.

Based on these considerations, accuracy is the performance metric used in the model classification and robustness. The attack's precision and recall were also considered because it is also an important measure to determine the attack's prediction rate between members and not members. Therefore, we define the best F-NIDS Gaussian hyperparameter being the instance that held the following indicators: binary accuracy greater than 80% and multi-class accuracy greater than 60%; attacks accuracy against real member inference attacks less than 65% and non-member inference attacks less than 40%, yielding an attacks performance 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.

Fig. 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 likelihood of success in the 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. However, it remains significantly greater then attack's accuracy baseline values, specially for the binary classification utility, it still maintains balanced utility between prediction and robustness levels as well. 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 4 presents some relevant results from the test performed such as attack baseline, attack membership accuracy, binary classification accuracy, and the F1 Score obtained by the attack.

Table 4
Performance of the rule-based adversarial attack.

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

Another important indicator is presented in Fig. 7b, which contains the precision over the obtained recall of the attack's results. Note that the precision of the attack's output is not significantly affected by increasing the level of noise. However, the recall has presented a decrease as the level of noise 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.

5.5. Robustness against attacks using an adversarial model

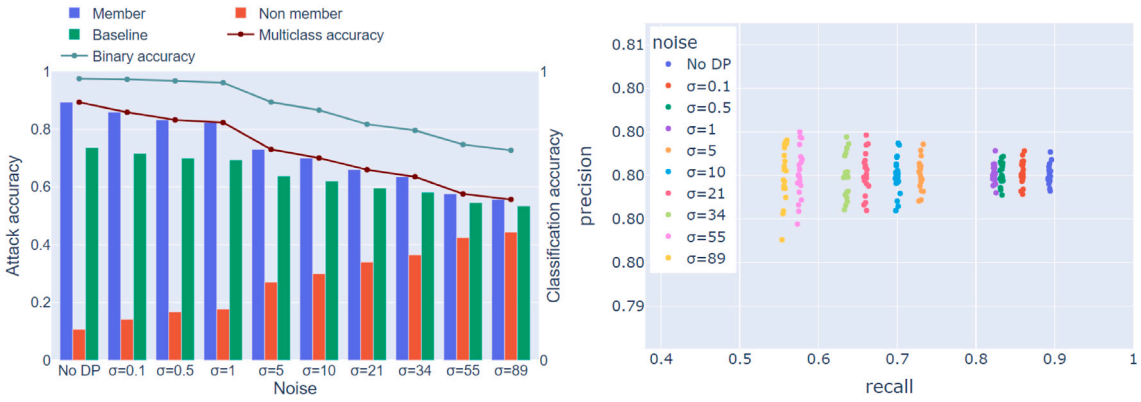
This of attack scenario is a kind of membership inference, but it can be done using just some dataset sample and previous knowledge about model architecture. A new kind of model can be trained and then turned into an adversarial model, called *ShadowModel*, capable of working as a membership predictor of entire dataset members, distributed across the other members, then compromising the entire dataset's privacy.

Although the classifier used by F-NIDS is protected with differential privacy, if a large sample is distributed over a few DAs, it can still represent a risk to be considered. This issue happens because, depending on its size, a sample may contain statistical properties similar to those of the original dataset. Federated learning, used in F-NIDS, plays an essential role by allowing the split of massive amounts of data into smaller samples as possible, and distributing it across larger quantities 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 DAs, to reduce as much as possible the risk of the stored sample 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 the 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 were 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 are shown in the Table 5. Each cycle was run with $100K$ member examples and $25K$ non-member examples.

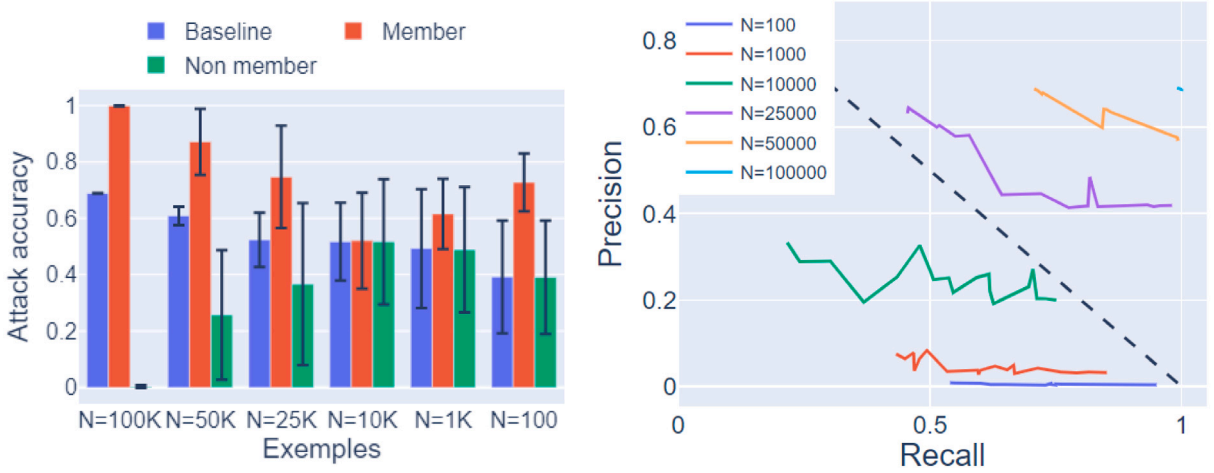
Fig. 8a presents the effectiveness of the attacks obtained by classifiers trained with different values of N. The vertical axis shows the percentage of attack success and horizontally the obtained results. It can be seen that the effectiveness in detecting members and non-members decreases as the value of N decreases. When this value of $N = 100$ is reached, it coincides with the lowest value recorded for

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



(a) Robustness and classification performances (b) Attack's precision and recall

Fig. 7. Accuracy and Robustness analysis for rule-based membership inference.



(a) Attack accuracy for each ShadowModel (b) Precision and Recall curves for each ShadowModel

Fig. 8. Accuracy and Robustness analysis for black-box membership inference.

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 example size would probably not be of much useful for intrusion detection. Therefore, a number of training examples $10K \leq N \leq 25K$ reach the reasonably acceptable level for the three indicators present in the figure, where both indicators have values close to 50% attack efficiency. Table 5 presents the obtained indicators, including the baseline, membership inference accuracy, and the F1 Score of the attack.

In Fig. 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, the 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 categorize non-members correctly.

Table 5
Attack's performance indicators using ShadowModels.

N	Baseline	Member	F1 Score adversarial
100 000	0.68 ± 0.00	0.99 ± 0.002	0,81 ± 0.001
50 000	0.60 ± 0.03	0.87 ± 0.12	0,72 ± 0.06
25 000	0.52 ± 0.1	0.74 ± 0.2	0,58 ± 0.11
10 000	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

5.6. Robustness against generative attacks using model inversion

The last robustness test performed on the F-NIDS classifier is the MI (Model Inversion). MI attacks are privacy attacks where an adversary tries to reconstruct sensitive information about individuals by exploiting the outputs of a machine learning model. In these attacks, the adversary uses queries to the model along with background knowledge to infer private attributes or data points that were used to generate the model's predictions [49]. By strategically querying the model and

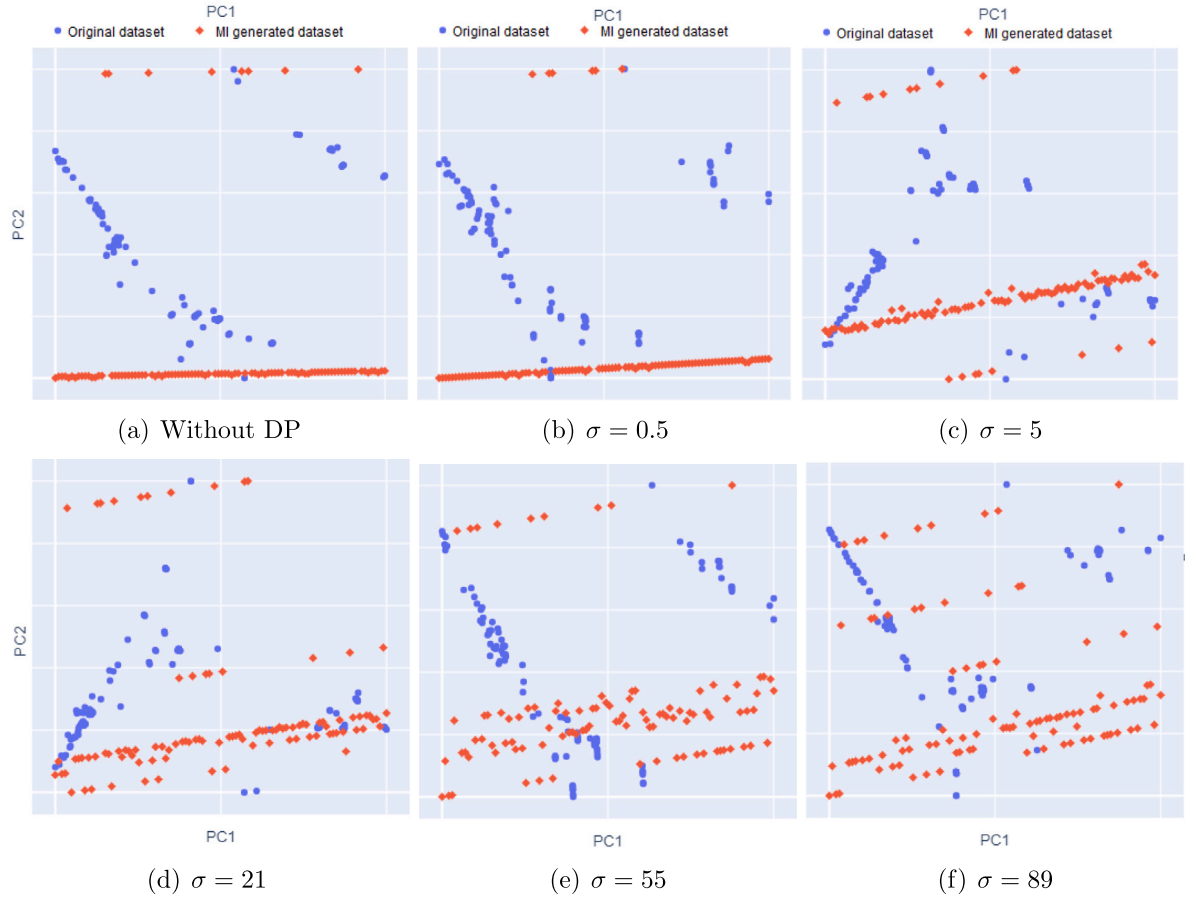


Fig. 9. Reduced dimensions to two components of the original and adversarial datasets.

analyzing its responses, the attacker can reverse-engineer sensitive information about individuals, potentially violating their privacy [50]. Model inversion attacks highlight the importance of protecting sensitive information and considering privacy-preserving techniques when developing and deploying machine learning models [51].

In a practical scenario, a DA needs to be hacked and its model unauthorized extracted. The malicious agent then uses the statistical properties that exist in the gradients to generate samples 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` [49], 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 5.4 sections. 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.

Fig. 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 Fig. 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. Another important aspect is that it is possible to draw the 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 Fig. 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 that fits both datasets. Another issue to note is that with $\sigma \geq 5$ present in Fig. 9c, it is no longer possible to draw the 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 too much distinguished from 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.

6. 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 were 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.

F-NIDS was designed to execute as a SaaS (Software as a Service) solution. Besides, the solution is to be industry agnostic, being able to work in multiple IoT networks. However, we strongly recommend using the system in privacy-critical environments, such as medical and financial domains, as it can significantly enhance data privacy while still offering acceptable classification performance.

Furthermore, the results allow us to define good settings to two hyperparameters which are very important to guarantee confidentiality while keeping the classification performance. These hyperparameters are the Gaussian noise σ and the 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 their own training data privacy ensured.

Although being the metrics used, in the F-NIDS's results, pretty similar to other related works to verify classification performance, in future works, other kinds of metrics will go to be investigated such as entropy, and avalanche effect. The system scalability and availability will also be investigated with appropriate metrics, such as load and stress indicators, response times, and error rate as well. Although confidentiality is a very important pillar of information security, data integrity, and availability are equally important factors to be reckoned with as well. Therefore F-NIDS needs to have metrics that approach these other pillars too. Keeping this in mind, this kind of scenario will be addressed in future works, other adversarial machine learning tests can be performed on classification models, to measure their robustness against other adversarial ML attack types. The other adversarial threats that can compromise the system classification model are evasion and poisoning. These attacks are related to the other mentioned security pillars and performing measurements, thus, they can help to improve the F-NIDS robustness not only in confidentiality but also in data integrity and availability.

Declaration of competing interest

Potential conflict of interest exists:

No conflict of interest exists.

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Data availability

Data will be made available on request.

Acknowledgments

This research was funded by the FAPESP (São Paulo Research Foundation), Brazil grant #2020/05126-6 and #2021/06210-3, FAPEMIG (Minas Gerais Research Foundation), Brazil grant #APQ-02675-21.

Funding

Funding was received for this work.

All of the sources of funding for the work described in this publication are acknowledged below:

FAPESP (São Paulo Research Foundation) grant #2020/05126-6 and #2021/06210-3, FAPEMIG (Minas Gerais Research Foundation) grant #APQ-02675-21.

Research ethics

We further confirm that any aspect of the work covered in this manuscript that has involved human patients has been conducted with the ethical approval of all relevant bodies and that such approvals are acknowledged within the manuscript.

IRB approval was obtained (required for studies and series of 3 or more cases)

Written consent to publish potentially identifying information, such as details or the case and photographs, was obtained from the patient(s) or their legal guardian(s).

References

[1] M.A. Rahman, A.T. Asyhari, The emergence of internet of things (IoT): Connecting anything, anywhere, Computers 8 (2) (2019) http://dx.doi.org/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 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 Soc. 50 (2019) 101660, http://dx.doi.org/10.1016/j.scs.2019.101660, URL https://www.sciencedirect.com/science/article/pii/S2210670718316883.

[4] J. Nayak, B. Naik, P.B. Dash, S. Vimal, S. Kadry, Hybrid Bayesian optimization hypertuned catboost approach for malicious access and anomaly detection in IoT nomalyframework, Sustain. Comput.: Inform. Syst. 36 (2022) 100805, http://dx.doi.org/10.1016/j.suscom.2022.100805, URL https://www.sciencedirect.com/science/article/pii/S2210537922001366.

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

[6] 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.

[7] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, P. Faruki, Network intrusion detection for IoT security based on learning techniques, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2671–2701, http://dx.doi.org/10.1109/COMST.2019.2896380.

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

[9] 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 Soc. 61 (2020) 102324, http://dx.doi.org/10.1016/j.scs.2020.102324, URL https://www.sciencedirect.com/science/article/pii/S221067072030545X.

[10] J. Cabrero-Holgueras, S. Pastrana, SoK: Privacy-preserving computation techniques for deep learning, Proc. Priv. Enhanc. Technol. 2021 (4) (2021) 139–162, http://dx.doi.org/10.2478/popets-2021-0064.

[11] 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, http://dx.doi.org/10.1109/SPW50608.2020.00028.

[12] 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, http://dx.doi.org/10.1109/DAC18072.2020.9218662.

[13] H. Zhu, H. Zhang, Y. Jin, From federated learning to federated neural architecture search: a survey, Complex Intell. Syst. 7 (2021) http://dx.doi.org/10.1007/s40747-020-00247-z.

[14] C.L. Stergiou, K.E. Psannis, B.B. Gupta, InFeMo: Flexible big data management through a federated cloud system, ACM Trans. Internet Technol. 22 (2) (2021) http://dx.doi.org/10.1145/3426972.

[15] G. Hohpe, B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley Longman Publishing Co., Inc., USA, 2003.

[16] C. Pautasso, O. Zimmermann, F. Leymann, Restful Web Services vs. "Big"' Web Services: Making the right architectural decision, in: Proceedings of the 17th International Conference on World Wide Web, WWW '08, Association for Computing Machinery, New York, NY, USA, 2008, pp. 805–814, http://dx.doi.org/10.1145/1367497.1367606.

[17] M. Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley Longman Publishing Co., Inc., USA, 2002.

[18] 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, in: Proceedings of Machine Learning Research, vol. 54, PMLR, 2017, pp. 1273–1282, URL https://proceedings.mlr.press/v54/mcmahan17a.html.

[19] K.A. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C.M. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T.V. Overveldt, D. Petrou, D. Ramage, J. Roselander, Towards federated learning at scale: System design, in: SysML 2019, 2019, URL https://arxiv.org/abs/1902.01046, (in press).

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

[21] J. Konečný, H.B. McMahan, F.X. Yu, P. Richtarik, A.T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, 2016, URL https://arxiv.org/abs/1610.05492.

[22] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019) http://dx.doi.org/10.1145/3298981.

[23] 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. Gardner, 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.

[24] 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, http://dx.doi.org/10.1109/SP.2017.41.

[25] 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.

[26] 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, http://dx.doi.org/10.1109/IMCOM53663.2022.9721785.

[27] Z. Ling, Z.J. Hao, An intrusion detection system based on normalized mutual information antibodies feature selection and adaptive quantum artificial immune system, Int. J. Semant. Web Inf. Syst. 18 (1) (2022) 1–25, http://dx.doi.org/10.4018/IJSWIS.308469.

[28] Z. Ling, Z.J. Hao, Intrusion detection using normalized mutual information feature selection and parallel quantum genetic algorithm, Int. J. Semant. Web Inf. Syst. 18 (1) (2022) 1–24, http://dx.doi.org/10.4018/IJSWIS.307324.

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

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

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

[32] R. M., M. Ahmed, R. Khan, An adaptive distributed intrusion detection system architecture using multi agents, Int. J. Electr. Comput. Eng. (IJECE) 9 (2019) 4951, http://dx.doi.org/10.11591/ijece.v9i6.pp4951-4960.

[33] G. Raja, S. Anbalagan, G. Vijayaraghavan, S. Theerthagiri, S.V. Suryanarayan, X.-W. Wu, SP-CIDS: Secure and private collaborative IDS for VANETs, IEEE Trans. Intell. Transp. Syst. 22 (7) (2021) 4385–4393, http://dx.doi.org/10.1109/TITS.2020.3036071.

[34] 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, http://dx.doi.org/10.1109/ACCESS.2021.3051530.

[35] B. Qian, J. Su, Z. Wen, D.N. Jha, Y. Li, Y. Guan, D. Puthal, P. James, R. Yang, A.Y. Zomaya, O. Rana, L. Wang, M. Koutny, R. Ranjan, Orchestrating the development lifecycle of machine learning-based IoT applications: A taxonomy and survey, ACM Comput. Surv. 53 (4) (2020) http://dx.doi.org/10.1145/3398020.

[36] A. Salem, Y. Sautter, M. Backes, M. Humbert, Y. Zhang, BAAAN: Backdoor attacks against autoencoder and GAN-based machine learning models, 2020.

[37] 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 Trans. Ind. Inform. (2021) 1, http://dx.doi.org/10.1109/TII.2021.3126728.

[38] A. Kundu, Fed+: A unified approach to robust personalized federated learning, 2021, http://dx.doi.org/10.13140/RG.2.2.27907.43040.

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

[40] 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, http://dx.doi.org/10.1109/CloudCom.2016.0085.

[41] 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 Things 19 (2022) 100538, http://dx.doi.org/10.1016/j.iot.2022.100538, URL https://www.sciencedirect.com/science/article/pii/S2542660522000403.

[42] Z. Bu, J. Dong, Q. Long, W. Su, Deep Learning With Gaussian Differential Privacy, Harvard Data Sci. Rev. 2 (3) (2020) https://hdsr.mitpress.mit.edu/pub/u24wj42y.

[43] M. Abadi, A. Chu, I. Goodfellow, H.B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 308–318, http://dx.doi.org/10.1145/2976749.2978318.

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

[45] H. Debar, J. Viinikka, Intrusion detection: Introduction to intrusion detection and security information management, in: Foundations of Security Analysis and Design III, Vol. 3655, 2005, pp. 207–236, http://dx.doi.org/10.1007/11554578_7.

[46] M. Sarhan, S. Layeghy, M. Portmann, Towards a standard feature set for network intrusion detection system datasets, Mob. Netw. Appl. 27 (1) (2022) 357–370, http://dx.doi.org/10.1007/s11036-021-01843-0.

[47] A. Narayanan, V. Shmatikov, Robust de-anonymization of large sparse datasets, in: 2008 IEEE Symposium on Security and Privacy (Sp 2008), 2008, pp. 111–125, http://dx.doi.org/10.1109/SP.2008.33.

[48] S.R. Ganta, S.P. Kasiviswanathan, A. Smith, Composition attacks and auxiliary information in data privacy, 2008, arXiv:0803.0032.

[49] 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, pp. 1322–1333, http://dx.doi.org/10.1145/2810103.2813677.

[50] B. Hitaj, G. Ateniese, F. Perez-Cruz, Deep models under the GAN: Information leakage from collaborative deep learning, 2017, arXiv:1702.07464.

[51] F. Tramèr, F. Zhang, A. Juels, M.K. Reiter, T. Ristenpart, Stealing machine learning models via prediction APIs, in: 25th USENIX Security Symposium (USENIX Security 16), USENIX Association, Austin, TX, 2016, pp. 601–618, URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer.

Jonathas Alves de Oliveira is a Master's Degree student at the Department of Electrical Engineering at the University of Brasília (UnB). He graduated in Computer Science in 2016 and has been working in the IT industry since 2008. Jonathas has experience in IT security, network projects, and software engineering. Currently, he holds the position of Software Architect at Dell Technologies. His areas of interest are Internet Security, IoT, smart home, and machine learning.

Vinícius P. Gonçalves has a Ph.D. in Computer Science and Computational Mathematics (2016) from the University of São Paulo (USP). He was also a research fellow at the University of Arizona (USA) before joining the University of Brasília (UnB). Dr. Gonçalves was a Postdoctoral Researcher at the USP Medical School, with a CAPES Fellowship. Currently, he is an Assistant Professor in the Electrical Engineering Department (ENE) at UnB, Brasília, Brazil, where he is a member of the Graduate Programs in Electrical Engineering (PPGEE and PPEE). Dr. Gonçalves is a researcher and member of the AQUARELA Group; his main research interests include Human–Computer Interaction, Internet of Things, Cybersecurity, Mobile Health, Image Processing and Machine Learning.

Rodolfo I. Meneguette is a professor at University of São Paulo (USP). He received his Bachelor's degree in Computer Science from the Paulista University (UNIP), Brazil, in 2006. He received his master's degree in 2009 from the Federal University of São Carlos (UFSCar). He received his doctorate from the University of Campinas (Unicamp), Brazil, in 2013. In 2017 he did his post-doctorate in the PARADISE Research Laboratory, University of Ottawa, Canada. His research interest are in the areas of vehicular networks, resources management, flow of mobility, and vehicular clouds.

Rafael Timóteo de Sousa Júnior WoS ResearcherID V3293-2019, holds a Bachelor's degree in Electrical Engineering from the Federal University of Paraíba - UFPB, Campina Grande, Brazil, in 1984. He obtained his Master's/DEA degree in Information Systems and Computing from Ecole Supérieure d'Electricité - Supélec, Rennes, France, in 1985, and his Ph.D. in Telecommunications and Signal Processing from the University of Rennes 1, Rennes, France, in 1988. He was a visiting researcher at the Network Security and Information Systems Research Group (SSIR) at Ecole Supérieure d'Electricité - Supélec, Rennes, France, from 2006 to 2007. He worked in the private sector from 1988 to 1996. His research focuses on cybersecurity, information and network security, distributed data services, machine learning for intrusion and fraud detection, as well as signal processing, energy harvesting, and physical layer security.

Daniel L. Guidoni, Department of Computer Science, Federal University of Ouro Preto, Brazil Daniel L. Guidoni received his Ph.D. degree in computer science from the Federal University of Minas Gerais, Belo Horizonte, Brazil, in 2011. He was awarded the best doctoral thesis in Brazil considering the computer science theses. He is currently an Associate Professor with the Federal University of Ouro Preto, Minas Gerais, Brazil. His research interests include wireless networks, vehicular networks, IoT, Smart Cities, and communication protocols.

José Carlos Martins Oliveira has a degree in Mathematics (1991), a Master's in Computer Science (2002) and Ph.D. in Industrial Engineering (2018). He is an adjunct professor in the field of Computer Science at the State University of Southwest Bahia-UESB, from 1997 to date. He is a professor and researcher in the areas of theory of computation, intelligent systems, weightless neural networks, granular computing, fuzzy logic, data science e machine learning. He has experience in administrative and academic management in basic and higher education. He currently coordinates, since 2019, the Center for Open and Distance Education at UESB, as well as the General Coordination of the Open University System of BrazilUAB/Capes/UESB.

Geraldo P. Rocha Filho is a Professor at the Department of Exact and Technological Sciences at the State University of Southwest Bahia (UESB). He was an effective Professor (2019–2022) at the computer science department at the University of Brasília (UnB). He was Researcher at the Institute of Computing at UNICAMP through the Post-Doctorate funded by FAPESP. He obtained the title of Doctor and Master in Computer Science and Computational Mathematics from ICMC-USP with a FAPESP scholarship. In the last five years, he has obtained 24+ publications in international journals and 32+ publications in conferences. His research interests are wireless sensor networks, vehicular networks, smart grids, smart home, and machine learning.