Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning?

Sawsan Abdul Rahman, Hanine Tout, Chamseddine Talhi, and Azzam Mourad

Abstract

With the ever increasing number of cyber-attacks, Internet of Things (IoT) devices are being exposed to serious malware, attacks, and malicious activities alongside their development. While past research has been focused on centralized intrusion detection assuming the existence of a central entity to store and perform analysis on data from all participant devices, these approaches cannot scale well with the fast growth of IoT connected devices and introduce a single-point failure risk that may compromise data privacy. Moreover, with data being widely spread across large networks of connected devices, decentralized computations are very much in need. In this context, we propose in this article a Federated Learning based scheme for IoT intrusion detection that maintains data privacy by performing local training and inference of detection models. In this scheme, not only privacy can be assured, but also devices can benefit from their peers' knowledge by communicating only their updates with a remote server that aggregates the latter and shares an improved detection model with participating devices. We perform thorough experiments on an NSL-KDD dataset to evaluate the efficiency of the proposed approach. Experimental results and empirical analysis explore the robustness and advantages of the proposed Federated Learning detection model by reaching an accuracy close to that of the centralized approach and outperforming the distributed unaggregated on-device trained models.

Introduction

As the name implies, Internet of Things (IoT) is about connecting a very wide variety of things to the Internet. Any object equipped with processor, network connectivity, actuators and embedded sensors can be part of the IoT environment. By connecting the devices and transferring the sensed data over the network, IoT provides useful services in numerous applications such as home automation, transportation systems, health, social life, agriculture, and many more [1]. IoT technologies push the connected devices to be monitored, engaging the collection and analysis of a huge amount of data, yet raising serious security issues. As massive data between the devices and the outside resources is exchanged at the network layer, intruders can compromise this layer and can easily target many IoT devices, which lack appropriate security defenses. Moreover, the ever increasing number of IoT devices lacking computation resources fails at security and opens opportunities for intruders to access them through different manners such as botnets with distributed denial of service, collusion attacks, malicious emails, and many more. Gemalto [2], the world leading company in digital security, highlighted that 52 percent of businesses are still not able to detect whether their IoT devices have been breached. When unknown cyber-attacks are increasingly emerging, and when the devices are highly vulnerable to malicious activities and intrusions, Intrusion Detection Systems (IDSs) [1] that monitor the network and detect malicious activities become vital to adopt.

Intrusion detection approaches can be classified as either signature or anomaly based. For the signature-based approach, attack rules or patterns, known as signatures, are predefined and stored for further analysis. By comparing certain data collected from the devices to the signatures, only known intrusions can be detected, which prevents the signature-based techniques from detecting zero-day attacks. On the other hand, anomaly-based methods build a model by studying the behavior of the normal samples through their features, and any deviation can be detected as suspicious action on the device. In these intrusion detection approaches, machine learning (ML) methods have been extensively adopted [3] with their success in developing intelligent systems. Existing centralized-based intrusion detection solutions imply training data generated by IoT devices either on the cloud [4], or in closer fog infrastructure [5, 6]. Typically, data gathered from end terminals is used for training and generating a model, which is used for prediction/classification. Although these approaches are able to detect intrusions with high accuracy, there are many issues entailed with such practices. First, with the massive amount of data generated by end-devices and communicated to the operator data center, latency is a significant issue that should be taken into consideration. This drives to longer processing time due to the distance between the IoT devices and the geographic location of the intrusion detection system. Considering a regular system having a single attack detector entity, the latter would perform badly when handling a large number of connected IoT devices, as it engages more processing data and time. Moreover, sending all data over the network imposes other problems such as high transfer cost and

Digital Object Identifier:
10.1109/MNET.011.2000286

Sawsan AbdulRahman, Hanine Tout, and Chamseddine Talhi are with École de Technologie Supérieure (ÉTS), Montreal; Azzam Mourad is with the Lebanese American University (LAU).

0890-8044/20/$25.00 © 2020 IEEE

significant communication overhead. Further, the data held or sensed on these devices is private in nature. Sharing it over the networks also makes it vulnerable to a variety of attacks causing critical consequences. Eventually, attackers may take control over IoT connected devices. Hence, whenever raw data leaves its hosts, privacy and security will be on the line. On-device learning [7], known also as self-learning, has been advanced as a potential solution through which both training and inference are performed locally on the devices based on their own data. Nonetheless, on-device learning is limited to per user experience independently. In other words, if the same application is running and the same behavior is detected on different devices, the latter will not benefit from each others' detection model as no knowledge is shared among them. Accordingly, both centralized and on-device learning still maintain crucial issues raising the need to rethink the design of machine learning, particularly the way data is communicated and analyzed. Further, such a solution should hold a trade-off between privacy, accuracy, communication cost, and latency. In other words, it has to preserve data privacy while still maintaining good accuracy for the generated models with low communication cost and acceptable latency.

To address the aforementioned limitations, we propose in this article a privacy-preserving Federated Learning (FL) scheme for IoT intrusion detection. Recent innovations have opened the door for FL [8] which is nowadays turning the corner and heading toward widespread adoption, being able to share peers' knowledge and maintain privacy. In our proposed approach, instead of engaging a centralized entity to which the devices send their data, we decentralize the machine learning tasks by moving the training of IDS models into the devices. In this context, (1) the learning and inference processes are performed on the devices, (2) data is never shared with external parties, and (3) only model parameters are communicated with a centralized entity responsible for disseminating global improvement of the model, making the latter better for all participants. The aim of the proposed scheme is reaching a detection accuracy as close as the one of centralized IDS, while at the same time maintaining privacy and security of sensed data and reducing communication overhead. The contributions of this work are summarized as follows:

- Building a novel privacy-preserving federated learning scheme for IoT intrusion detection, which trains the models on devices and federates their learning. We believe that the provided investigation of the literature and proposed approach may offer relevant insights for further research advancements in this direction.
- Elaborating thorough evaluation using the NSL-KDD dataset [9] and covering different real-life scenarios of data distribution with respect to the attack types. Experimental results explore the efficiency of sharing and aggregating the models in FL settings compared to the centralized and self-learning approaches (i.e., "Cloud-centric ML for Mobile" and "On-Device Learning" as presented in [7]).

The rest of the article is organized as follows. We present in the next section the various existing works for intrusion detection in IoT, as well as the approaches for FL. Then, we propose a privacy-preserving federated learning scheme for IoT intrusion detection, followed by a performance evaluation. Finally, the last section concludes the article.

Federated Learning and IoT Intrusion Detection: Literature Overview

In this section, we provide a state of the art review of both the IoT intrusion detection approaches and the different federated learning aspects.

IoT Intrusion Detection

With the development of IoT and the increasing number of their cyber-attacks, many approaches have been proposed for IoT intrusion detection. A framework to detect anomalies in IoT networks and to provide Security as a Service has been presented in [1]. The framework consists of three phases: network connection, anomaly detection, and mitigation. In the first phase, the host network is monitored, and the network protocol currently used is identified in order to make a virtual network connection. Next, a machine learning module is dedicated for anomaly detection. This module allows optimized data packet collection and transformation, which enables IDS to run on resource constrained hardware. In the last phase, an actuator module raises a mitigation flag and the handler component reacts to the response accordingly.

An anomaly detection system is presented in [10] where a dimension reduction model and a classifier are deployed. For dimension reduction, Principle Component Analysis is used to prevent both performance degradation of fault diagnosis and real-time requirement threat caused by complex computations. As for the classifier, the softmax regression algorithm is leveraged and compared to the k-nearest neighbor (knn) when varying the number of features from 3, 6 to 10. In [3], the authors have proposed a classification algorithm for intrusion detection while considering the resource-constraint devices of IoT. The solution encompasses two distinct phases. First, a Negative Selection algorithm that is robust to complex classification problems is used. Since this algorithm is unsuitable for large-scale classification of normal and abnormal samples, it is implemented to build a training set representing only the normal behavior of the network. Then, a Neural Network is trained and used for the actual classification of the attack samples. Thus, this approach does not impose the overhead of the training process on the end-devices and conserves their limited power and computational capabilities.

The IoT framework, implemented in [5] for attack detection, is based on fog computing. As opposed to the centralized mechanisms that poorly perform due to the IoT device requirements such as low latency, resource constraints, and scalability, fog computing replaces cloud computing in this approach. The attack detection of a group of devices is handled by the fog to which they are connected, where an ELM-

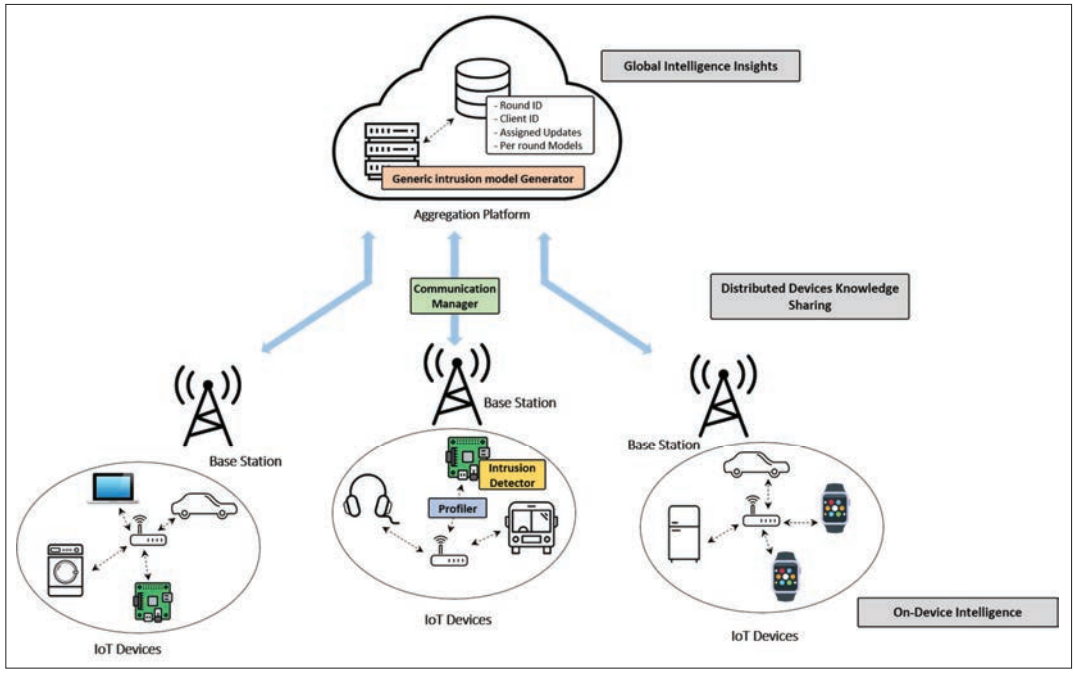

FIGURE 1. High level architecture: Federated Learning for IoT intrusion detection.

based Semi-Supervised Fuzzy C-Means method is performed. In [6], a distributed based attack detection mechanism has been proposed while considering the underlying requirements of IoT devices such as resource restrictions, low power and mobility. The attack detection in the proposed method, which is performed using Deep Learning, is done at the fog layer where the models are trained on a distributed number of nodes.

In the aforementioned approaches, the training model and intrusion detection methods are performed on central nodes, whereas different approaches are applied in which a centralized server/cloud/fog takes responsibility of training and attack detection. However, this process demands transmitting IoT data outside the devices, which does not ensure privacy, risks the security of transmitted data, entails high latency overhead, and requires significant communication cost.

Federated Learning

Recent research works in Federated Learning have been advanced to address different aspects of communication cost, client selection, privacy, security and resource allocation. In what follows, we present an overview of these aspects. As FL requires that each user downloads the current version of the model and then sends it to the cloud after updating it, a new problem related to connectivity arises, especially because the uplink has generally lower network connection compared to the downlink. To minimize the communication cost, the authors in [11] have proposed to replace the enormous number of rounds required in the typical FL with only one. Such a one-shot FL approach requires the clients to complete the training until convergence, and only the one having the baseline amount of data can share its model updates. The client selection aspect in Federated Learning has been discussed in [12]. As longer update and upload time of the models are engaged when resource constrained devices are participating, the proposed approach selects the clients according to their time-based resources. Moreover, by providing some incentives, another set of clients is selected to upload a small amount of their raw data to the server in order to make the FL model more robust.

Although Federated Learning has been proposed as a privacy-preserved solution, malicious actors may still find their way into such systems. Therefore, the authors in [13] have proposed a new privacy-preserved scheme to detect causative attacks, which are able to harm the learned model by injecting malicious training results. The integrity of the deep learning training process is guaranteed based on a trusted execution environment. In [14], a secure mechanism for data collaboration is built in the IoT environment. Under the setting of Federated Learning, the framework deals with large-scale data from multiple parties and their data security is guaranteed using the Blockchain paradigm.

Special attention has been given to study FL resource allocation in the context of wireless networks. In [15], the authors have proposed a model for analyzing and characterizing the performance of FL. Tractable expressions are derived for the convergence rate of FL considering the effects of both scheduling schemes and intercell interference. Moreover, they have studied the effectiveness (convergence rate) of random scheduling, round robin, and proportional fair scheduling policies.

While FL is currently an active research area, many efforts are being advanced for each of the discussed aspects. However, there is still lots of room for further FL-based contributions in different fields. Along with this wave of innovations, we aim in this work to investigate the applicability and efficiency of federated learning for IoT intrusion detection.

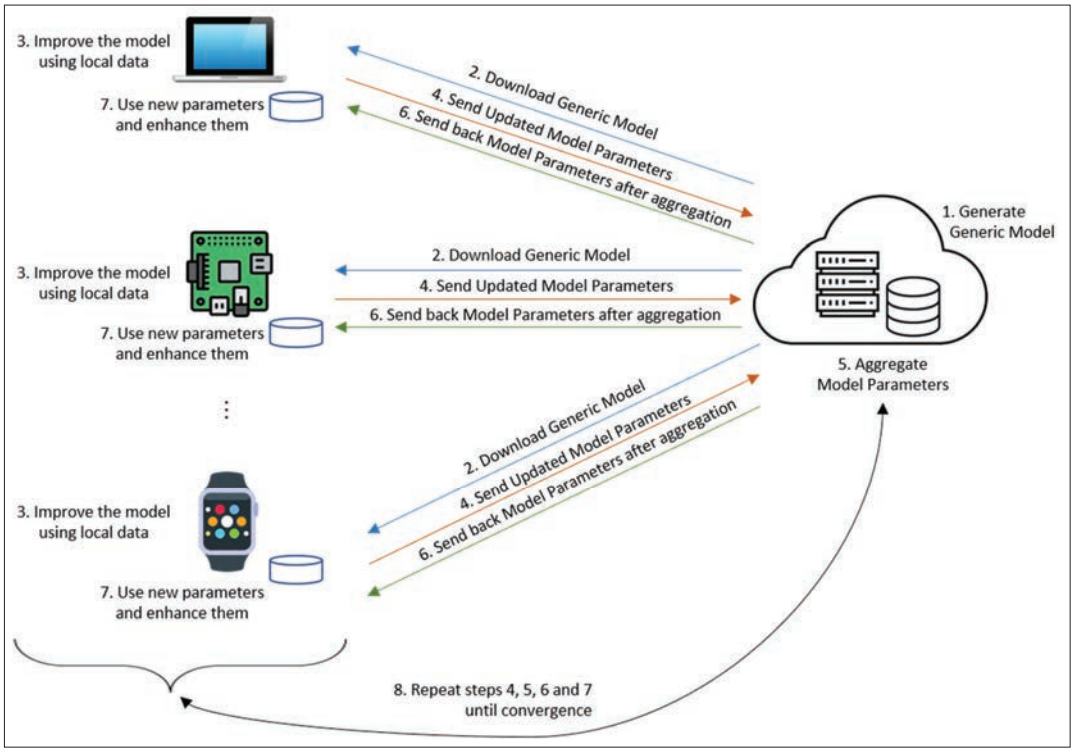

FIGURE 2. Illustrative scenario: interactions among the different modules for achieving federated intrusion detection.

Federated Learning for IoT Intrusion Detection: A New Privacy-Preserving Scheme

Figure 1 depicts the architecture of the proposed FL scheme for IoT intrusion detection, where a large number of devices are connected to the network and positioned in different sites. While these devices are connected to gateways, their network data is monitored by a profiler for further analysis in the AI-based intrusion detector running on IoT devices. Such on-device intelligence offers the autonomy of local intrusion detection through local execution of the training, parameters optimization and inference. Further, it accelerates the detection time as analysis is performed locally where the data is generated. Selected IoT devices communicate their machine learning models with a server-based aggregation platform, which is in place to aggregate their models and generate an enhanced intrusion detection model with optimized parameters. All the communications between the IoT devices and the aggregation platform are handled by the communication manager. Adapting existing relevant solutions (e.g., [12]) for selecting the clients participating in the FL process should take into consideration the IoT devices resources, time consumption, and communication cost. Exchanging the detection models with the server-based platform provides global intelligence insights on the latter, aiming to achieve close accuracy of centralized ML analysis, which rather has global data insights. The optimized model is then communicated back with the distributed IoT devices, and hence the knowledge is shared among them. This sharing scheme results in better learning as it enables a device to detect intrusions learned based on comparable behavior generated from different participating devices.

An illustrative scenario exploring the interactions among the different modules of the proposed FL IoT intrusion detection scheme is depicted in Fig. 2. Initially, random IoT nodes, which are idle, charging, and on unmetered WiFi connection, are chosen by the server to participate in the FL process. Afterwards, the different parties of the system communicate as follows.

Step 1: The server generates a generic intrusion model where a neural network architecture is built. At this stage, the number of hidden layers, neurons, epochs, and so on, are identified.

Step 2: The model is downloaded by the nodes that wish to use the model, whether contributing or not in the FL process.

Step 3: The selected nodes keep their local data private and use them on-device to enhance the model being studied.

Step 4: Only the model parameters of the updated intrusion detection model are shared with the central server instead of sending sensitive data and intruding the privacy of the nodes.

Step 5: The server aggregates the weights from the different node models once all the updates are received and creates a new updated model. For the aggregation, the FederatedAveraging algorithm is used, in which the parameters are weighted based on the nodes dataset size. The full process and steps of the algorithm are presented in [8].

Step 6: The server pushes back the updated model parameters to the nodes.

Step 7: Each node uses the updated model parameters and improves them based on its new generated data.

Steps 4, 5, 6, and 7 are repeated for continuous learning and improvement of the model.

The added value of federated intrusion detection models lies in the following. By replacing the

traditional machine learning approaches with FL, security and privacy of data generated by IoT devices are preserved while reducing the communication overhead. The high amount of sensitive and private data is not shared anymore with a central server. Instead, the training is done locally on the devices and only the model updates are exchanged. In addition, FL not only gets rid of the latency that occurs from the large amount of transmitted data, but also allows real time prediction of the anomalies with on-device prediction. Moreover, when there is no connection, the devices can still independently predict and detect anomalies in the network since the models are locally presented. Compared also with on-device self-learning, FL gets the benefit of the peers' model and makes it possible to detect intrusions not generated previously by its own traffic.

Evaluation Methodology

We provide in this section a brief description of the dataset used in our experiments, the pre-processing phase performed on the data, and the metric used for evaluation.

Data Description

NSL-KDD [9] is a widely adopted dataset for the evaluation of intrusion detection methods, which we have used to validate our model. The dataset contains 148,517 records split for the training and testing, each of 41 features such as duration, protocol type, service, flag, and so on. The benign records are labeled as normal, while the malicious ones are labeled based on the specific attack types falling in one of the following categories: Denial of Service (DoS), User to Root Attack (U2R), Remote to Local Attack (R2L) and Probing Attack. Beside the attack types presented in the training set, the test set includes 17 new attack types to make the data more realistic and evaluate the efficiency of the models in detecting unknown attacks. From the NSL-KDD dataset, we have used in our experiments the 'KDDTrain+' dataset for data distribution over the nodes and models training, and the 'KDDTest+', which is placed and performed on each node device for testing.

Data Pre-Processing

Before starting training the models, data has been prepared for consumption. First, we have encoded the three categorical features; "protocol_type," "service," and "flag" using one-hot encoding. The latter converts the data to numerical values, to be handled by the neural network. Next, a distance-based method has been adopted as the features in the NSL-KDD are distributed very widely, which results in domination of some features and missing out of critical information in others. In our experiments, we have used Min-Max normalization to scale our features to a 0–1 range. We have used the same pre-processing techniques for testing. When performing the encoding technique, we have made sure that both training and testing sets are consistent in terms of number of columns generated. As for the normalization, proper scaler between the two sets is used to avoid bias in the results. After tuning the hyper-parameters, we have built the FL model using 122 input variables, 288 neurons for the hidden layer, and two neurons in the output layer to represent the abnormal and normal decision. The same configuration has been applied for both the centralized and self-learning techniques for fair comparison.

Evaluation Criteria

The following are the common indicators used to analyze the intrusion detection performance:

- True Positive (TP): Indicates the number of abnormal samples correctly classified as abnormal.
- True Negative (TN): Indicates the number of normal samples correctly classified as normal.
- False Positive (FP): Indicates the number of normal samples incorrectly classified as abnormal.
- False Negative (FN): Indicates the number of abnormal samples incorrectly classified as normal.

Based on the aforementioned indicators and their confusion matrix, we have adopted the accuracy metric to evaluate and compare the generated models.

Experimental Results: Centralized vs On-Device vs Federated Learning

The objectives of our experiments, conducted using Raspberry Pi devices, are the following:

- Build and evaluate a centralized model, where the full training set is stored and processed on the server.
- Distribute the training data over nodes while evaluating how efficient is each node's model and how close is compared to the centralized model. This represents the self-learning setting, where each node trains its own data only without benefiting from peer's data or models.
- Embed and evaluate the federated learning scheme using five rounds of communication. In these experiments, we analyze the improvement of the models after being aggregated and we compare as well the FL models to the centralized and self-learning (i.e., "Cloud-centric ML for Mobile" and "On-Device Learning" as presented in [7]) approaches. The centralized-based scenario is considered the best in terms of model accuracy, yet with major privacy issues and overhead of data communications since training data is all gathered on a remote entity. Therefore, the provided experiments explore that our federated learning-based scheme can reach comparable accuracy while guaranteeing data privacy.

In the centralized approach, 100 percent of the data is used for training and testing. Using the obtained confusion matrix (TP = 9 308, TN = 9 423, FP = 288, and FN = 3 525), the results show 83.09 percent accuracy.

Use Case #1: Data Distribution Per Attack Type

In this set of experiments, we consider the use case where four nodes are distributed over the network, each representing an intrusion detection system. We assume that the 4 IDS monitoring network traffic are generated by 4 sub-networks,

	Round #1	Round #2	Round #3	Round #4	Round #5
Node #1	9,185 Dos + 9,185 N	18,370 Dos + 18,370 N	27,555 Dos + 27,555 N	36,740 Dos + 36,740 N	45,927 Dos + 45,927 N
Node #2	2,331 P + 2,331 N	4,662 P + 4,662 N	6,993 P + 6,993 N	9,324 P + 9,324 N	11,656 P + 11,656 N
Node #3	199 R2L newline + 199 N	398 R2L newline + 398 N	597 R2L newline + 597 N	796 R2L newline + 796 N	995 R2L newline + 995 N
Node #4	10 U2R newline + 10 N	20 U2R newline + 20 N	30 U2R newline + 30 N	40 U2R newline + 40 N	52 U2R newline + 52 N

TABLE 1. Training data distribution for use case #1.

where the malicious traffic seen by each IDS belongs to one specific attack type. Therefore, nodes 1, 2, 3, and 4 represent IDS monitoring network traffic of Dos, Probe, U2R, and R2L, respectively. Besides the abnormal data, normal samples of the same number of the attacks are randomly distributed among the clients where no redundant data exists. Accordingly, Table 1 shows the training data distribution of the 4 IDS. Each round represents at certain time the nodes' data, which is equally distributed among rounds and accumulated from one round to another. This data accumulation simulates the real behavior of these nodes, which generate new data over several rounds.

Figure 3a depicts the results for use case #1. Considering the first node, we can see that its accuracy ranges from 73.34 percent in round 1 to 75.16 percent in round 5 of the self-learning setting, and from 74.06 percent in round 1 to 76.27 percent in round 5 in the case of FL. There is slight increase in the accuracy results of both approaches since the models corresponding to the first node are trained on enough DoS and normal samples, which represent around 76 percent of the test set. However, we can see that the FL outperforms the self-learning in the five rounds since the models are taking advantage of the models aggregation. The second node has the same interpretation as the first one, with more focus on the advantage of FL by reaching 75.06 percent accuracy in its last round compared to 70.7 percent for the self-learning. Moving to the results analysis of nodes 3 and 4, we can clearly see how the nodes are benefiting from their peers' models. For self-learning, the accuracy for node 3 reaches a maximum percentage of 49.99 percent in round 5, while node 4 reaches a maximum value of 52.24 percent. This can be explained by the amount of data used for training and their insignificant representation in the test set. However, for the same data distribution, node 3 is able to reach 76.06 percent accuracy from the second round when applying FL. As for node 4, the accuracy reaches 75.63 percent, 77.57 percent, 80.18 percent, and 82.23 percent for rounds 2, 3, 4, and 5, respectively. Based on the conducted experiments, we can also achieve our objective of showing how close all the accuracy values in the last FL round are compared to the central model, which is not the case in self-learning.

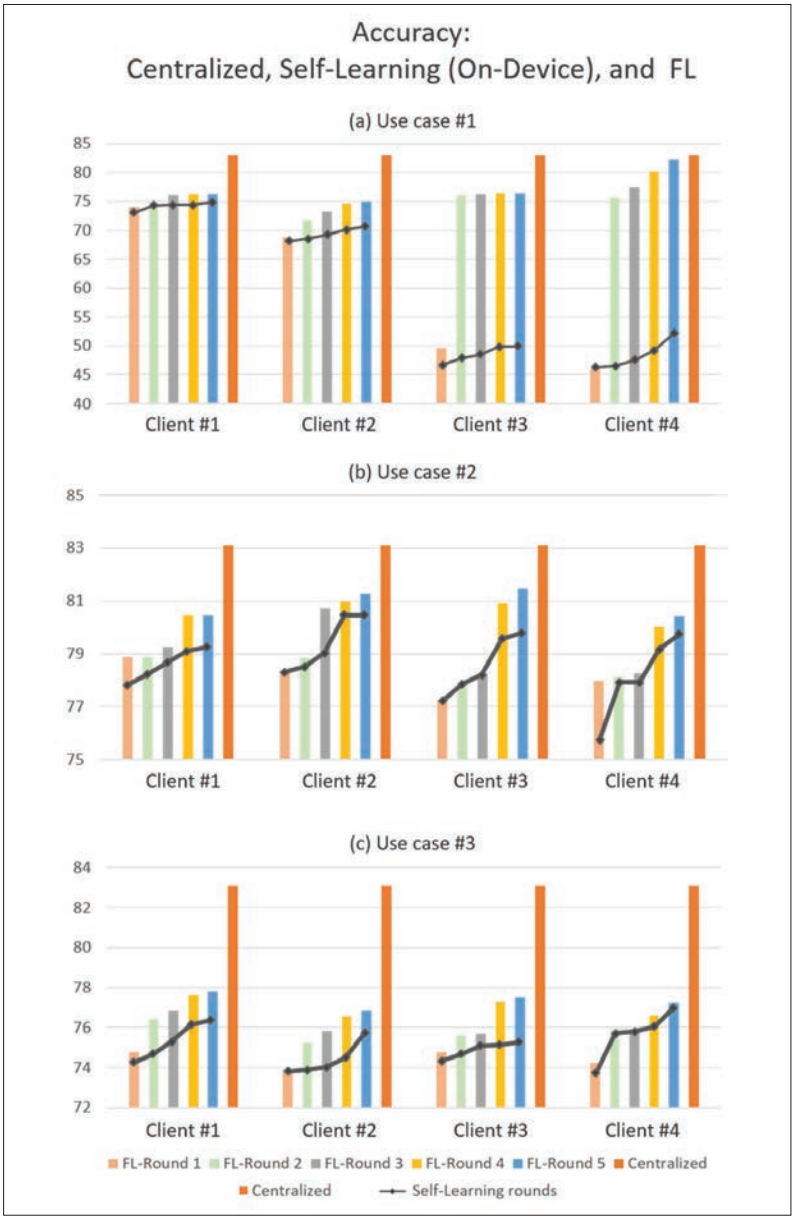

FIGURE 3. Comparison between centralized, self-learning (on-device), and FL for use cases #1, #2 and #3.

Use Case #2: Equal Data Distribution of Attack Types

To validate our solution and achieve our objectives in other scenarios, we studied the case where the 4 IDS distributed over the network are deployed to examine traffic coming from all the attack types. In such a scenario, DoS, Probe, U2R, and R2L are equally distributed among the 4 nodes, then equally distributed from one round to another while accumulating data from previous rounds. Each distribution contains normal traffic equal to the attack samples, not to bias the model toward one class. Table 2 shows the number of training samples for use case #2. More specifically, each node has 2296 samples of DoS, 582 samples of Probe, 49 samples of R2L, two samples of U2R, and 2,929 samples of Normal traffic in the first round. Then, these samples are accumulated at each round from 2 to 5.

Rounds 1 - 5	
Nodes	2,296 Dos + 582 Probe +
1 - 4	49 R2L + 2 U2R + 2,929 Normal

TABLE 2. Training data distribution for use case #2.

Round 5	
Node 1	12,860 Dos + 3,730 Probe + 269 R2L + 15 U2R + 16,865 normal
Node 2	8,725 Dos + 1,631 Probe + 176 R2L + hspace{1em} 9 U2R + 10,546 normal
Node 3	13,779 Dos + 4,429 Probe + 268 R2L + 17 U2R + 18,484 normal
Node 4	10,563 Dos + 1,866 Probe + 282 R2L + 11 U2R + 12,735 normal

TABLE3. Training data distribution in the last round for use case #3.

The results for the second use case studied are depicted in Fig. 3b. In self-learning, the nodes start in the first round with accuracy values ranging between 75.74 percent and 78.49 percent. The models are afterwards enhanced from one round to another until achieving between 79.24 percent and 80.47 percent accuracy. As for FL, the models are better and close to the central model in the last round where the accuracy ranges between 80.45 percent and 81.48 percent. This scenario is considered the ideal in terms of closeness to the centralized data distribution where each node trains on all traffic types and is able to predict their behaviors.

Use Case #3: Random Data Distribution of Attack Types

Another set of experiments has been taken into consideration where the samples used in the first two scenarios are randomly distributed among the nodes and rounds. As we simulate that the nodes are generating more data over time, we always consider the case of data accumulation from previous rounds. Also, we make sure that the number of normal and attack samples is consistent to avoid overfitting when building the models. We show in Table 3 the number of samples each node has in the last round, as in all the previous rounds it is a random distribution as well.

The results shown in Fig. 3c for the use case #3 reaffirm the efficiency of the proposed scheme with respect to other approaches. First, we can see how FL outperforms self-learning for all the models that have been built. For example, node #3 shows accuracy between 74.33 percent and 75.27 percent in the self-learning setting, and between 74.76 percent and 77.51 percent for FL. Moreover, the accuracy achieved in the last round of FL training, ranging from 76.84 percent to 77.79 percent, is always close to the centralized one.

Conclusion and Future Work

We proposed in this article a federated machine learning based intrusion detection scheme for IoT, which leaves data generated on-devices, trains their own models in order to maintain privacy and secure sensed data. To benefit from peers' models, a server aggregates the updates locally computed whenever a training round in Federated Learning from different devices is performed. The experimental evaluation showed that, after the last round of FL, the aggregated models have an accuracy fluctuating around 83.09 percent, which refers to a centralized model being trained over the entire dataset. Moreover, we compared the FL settings when sharing the models updates with a server, and without sharing them in a self-learning setting. The results showed that FL outperforms self-learning in all training rounds for all the studied use cases. Accordingly, we can conclude the following: (1) federated intrusion detection could reach comparable accuracy with respect to the centralized approach that has global insights of the overall system; (2) the knowledge aggregation in federated intrusion detection gave the latter the advantage to always outperform the self-learning approach. On the other hand, the selection of clients in FL has significant impact on the performance of the system. Devices dropout during the process, long response time for the upload and update of the models, clients with less relevant data, and so on, are just some of the side effects of existing approaches. Addressing these limitations, which drastically decrease the global model accuracy, is of great importance for future research directions.

References

[1] S. Chawla *et al.*, "Security as a Service: Real-time Intrusion Detection in Internet of Things," *Proc. 5th Cybersecurity Symposium*, ACM, 2018, p. 12.

[2] Gemalto, "Almost Half of Companies Still Can't Detect IoT Device Breaches," accessed: 2020-05-27; available: https://www:gemalto:com/press/pages/almost-half-of-companies-still-can-t-detect-iot-devicebreaches- reveals-gemalto-study:aspx.

[3] M. E. Pamukov, V. K. Poulkov, and V. A. Shterev, "Negative Selection and Neural Network Based Algorithm for Intrusion Detection in IoT," *Proc. 41st Int'l. Conf. Telecommunications and Signal Processing (TSP)*, IEEE, 2018, pp. 1–5.

[4] C. Thota *et al.*, "Centralized Fog Computing Security Platform for Iot and Cloud in Healthcare System," *Fog Computing: Breakthroughs in Research and Practic*, IGI Global, 2018, pp. 365–78.

[5] S. Rathore and J. H. Park, "Semi-Supervised Learning Based Distributed Attack Detection Framework for IoT," *Applied Soft Computing*, vol. 72, 2018, pp. 79–89.

[6] A. A. Diro and N. Chilamkurti, "Distributed Attack Detection Scheme Using Deep Learning Approach for Internet of Things," *Future Generation Computer Systems*, vol. 82, 2018, pp. 761–68.

[7] J. Konecny, "Privacy-Preserving Collaborative Machine Learning without Centralized Training Data," accessed: 2020-05-27; available: http://jakubkonecny:com/files/2018-01 UW Federated Learning:pdf, 2018.

[8] H. B. McMahan *et al.*, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *Proc. 20th Int'l. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2016.

[9] M. Tavallaee *et al.*, "A Detailed Analysis of the KDD Cup 99 Data Set," *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, IEEE, 2009, pp. 1–6.

[10] S. Zhao *et al.*, "A Dimension Reduction Model and Classifier for Anomaly-Based Intrusion Detection in Internet of Things," *Proc. 15th Int'l. Conf. Dependable, Autonomic and Secure Computing, 15th Int'l. Conf. Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, IEEE, 2017, pp. 836–43.

[11] N. Guha, A. Talwlkar, and V. Smith, "One-Shot Federated Learning," arXiv preprint arXiv:1902.11175, 2019.

[12] N. Yoshida *et al.*, "Hybrid-FL: Cooperative Learning Mechanism Using Non-IID Data in Wireless Networks," arXiv preprint arXiv:1905.07210, 2019.

[13] Y. Chen *et al.*, "A Training-Integrity Privacy-Preserving Federated Learning Scheme with Trusted Execution Environment," *Information Sciences*, vol. 522, 2020, pp. 69–79.

[14] B. Yin *et al.*, "FDC: A Secure Federated Deep Learning Mechanism for Data Collaborations in the Internet of Things," *IEEE Internet of Things J.*, 2020.

[15] H. H. Yang *et al.*, "Scheduling Policies for Federated Learning in Wireless Networks," *IEEE Trans. Commun.*, vol. 68, no. 1, 2020, pp. 317–33.

Biographies

Sawsan Abdul Rahman received the Masters degree in computer science from the Lebanese American University (LAU), Lebanon. She is a Ph.D. candidate at École de Technologie Supérieure (ÉTS), Montreal, Canada, working in the areas of AI, machine learning, and security. She is a reviewer for several prestigious conferences.

Hanine Tout received the Ph.D. degree in software engineering from École de Technologie Supérieure (ÉTS), Montreal, Canada. She is a postdoc fellow between ETS and Ericsson, Canada, where she is leading two industrial projects in the areas of AI, federated learning, machine learning, security, 5G and cloud-native IMS. She is a TPC member and reviewer of prestigious conferences and journals.

Chamseddine Talhi received the Ph.D. degree in computer science from Laval University, Quebec, Canada. He is an associate professor in the Department of Software Engineering and IT at ÉTS, University of Quebec, Montreal, Canada. He is leading a research group that investigates smartphone, embedded systems and IoT security. His research interests include cloud security and secure sharing of embedded systems.

Azzam Mourad is currently an associate professor of computer science at the Lebanese American University, and also an affiliate associate professor in the Software Engineering and IT Department, École de Technologie Superieure (ÉTS), Montreal, Canada. He has served/serves as an associate editor for *IEEE Transaction on Network and Service Management*, *IEEE Network*, *IEEE Open Journal of the Communications Society*, *IET Quantum Communication*, and *IEEE Communications Letters*. He has also served/serves as General Chair of IWCMC2020, General Co-Chair of WiMob2016, and Track Chair, TPC member, and reviewer of several prestigious journals and conferences. He is an IEEE senior member.