# Robust Distributed Intrusion Detection System for Edge of Things

Wassila Lalouani and Mohamed Younis
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County,
Baltimore, Maryland, USA,
{lwassil1, younis}@umbc.edu

***Abstract***— The edge computing paradigm has been adopted in many Internet-of-Things (IoT) applications to improve responsiveness and conserve communication resources. However, such high agility and efficiency come with increased cyber threats. Intrusion detection systems (IDS) have been the primary means for guarding networked computing assets against hacking attempts. The popular design methodology for IDS relies on the application of machine learning (ML) techniques that use intelligence data to classify malicious activities. However, in the realm of IoT, insufficient data is available to build IDS; hence a distributed intrusion system with continual data collection is primordial to refine the detection model. Such IDS is also subject to privacy constraints and should sustain robustness against data manipulation from internal attackers that degrade the ML model. This paper opts to fulfill these requirements by proposing a novel distributed IDS for IoT. The proposed system employs federated learning to enable privacy preservation and diminish the communication overhead. Our system promotes a reinforcement mechanism to ensure resiliency to data manipulation attacks by single or colluding internal actors. The validation results using recently released datasets demonstrate the effectiveness of our approach.

***Keywords: Federated learning, Fog computing, IoT, intrusion detection, collusive attacks.***

## I. Introduction

IoT interconnects numerous resource-constrained electronic devices and enables panoply of applications. Access to cloud computational and storage resources is often necessary to IoT end users. However, offloading IoT data to the cloud stresses the communication network and increases latency. Edge computing has emerged as an effective methodology to mitigate the effect of transferring voluminous data to the cloud. Basically, some data processing is conducted by nodes at the network edge such as smartphones, road side units, etc. Such architecture is referred to as Edge of Things (EoT) and may also involve fog nodes for increased local processing and consequently higher responsiveness to real-time applications. For example in a smart city application, traffic intensity can be assessed using on-road sensors and the cellphones of individual drivers; the collected data can be immediately processed at an edge node and based on the vehicle density, traffic signal timing could be adjusted for travel delay and vehicular throughput.

Given the role played by EoT, it is important to safeguard the edge devices against cyber-attacks that opt to manipulate the data and indirectly degrade the IoT utility. For EoT, intrusion detection through ML models is the most practical approach due to the relatively sparse intelligence database, given the recent emergence of IoT applications [1]. Therefore, edge devices have to share security logs and alerts in order to upgrade the capabilities of the IDS. However, a number of challenges arise. First, if the IDS is maintained in the cloud, the volume of the data sent by the edge nodes can potentially overwhelm the communication infrastructure. Second, the formed data-driven IDS models are highly dependent on the data quality. Malicious or erroneous data can be detrimental for the model accuracy and consequently the decision making process. Third, an owner of a device, e.g., a smartphone, will be concerned about privacy and reluctant to show what vulnerabilities are uncovered and what exploitations have succeeded or attempted.

To overcome the aforementioned challenges, this paper promotes a novel distributed IDS for EoT. The design exploits the advantages of the federated learning (FL) methodology in order to tackle the scalability and privacy concerns. FL can be viewed as a means for forming a global ML model out of multiple local data-driven models. Thus, instead of sharing alerts and security logs, an edge node will form a local IDS model and send the model parameters to an aggregator node. The latter federates the models of the individual edge devices and produces a more accurate IDS for the EoT nodes to use. The process is iterative and continual, where an upgraded IDS will be periodically generated based on the more recent local models transmitted by the edge devices. The proposed FL data processing architecture employs fog nodes and in essence could constitute a multi-tier aggregation tree rooted at the cloud. Our FL-based IDS (FLIDS) design also thwarts data poisoning attacks by individual and multiple colluding actors. Using a hierarchical architecture, we promote an insider attack detection mechanism that uses a reinforcement strategy to flag malicious actors. The validation results using a prominent dataset confirm the effectiveness of FLIDS.

The paper is organized as follows. The next section covers the related work. Section III discusses the system and attack models. Section IV presents FLIDS. Section V reports the validation results. Finally, the paper is concluded in Section VI.

## II. Related Work

Prior work on network intrusion detection focuses mainly on employing ML techniques in order to extract relevant features from security logs and detect malicious behaviors. Existing methodologies can be categorized as signature-based and anomaly-based [2]. Given the contribution, we focus on work on anomaly based IDS for IoT. Diverse conventional ML algorithms have been employed in the literature including, e.g., Naive Bayes, and SVM [3], as well as deep learning techniques such as LSTM [4]; however, the scope is limited to attack detection and not distinguishing malicious behaviors. Different datasets have been explored to form a universal IDS using diverse classes of anomalous behavior like, DoS, worm, etc. [1][5]. Nkiama et al. [6] have proposed a feature selection mechanism to exclude features that do not contribute to the detection rate. Meanwhile, Eskandari et al. [7] have focused on lightweight implementation of IDS on edge nodes. All the aforementioned studies do not consider the topology of EoT.

Distributed IDS have also been investigated. HADL [8] combines both ML and deep learning methods, where the latter is applied on the cloud to detect anomalies. Then, ML is executed on-demand on the level of an aggregation path to identify the suspected node. To sustain privacy, Tabassum et al. [9] pursue incremental learning, while Al-Athba et al. [10] employ federated learning. However, the aforementioned studies do not consider the presence of internal attacks. Although DIDS [11] handles internal attacks, the analysis is effectively centralized in nature using the raw data, and hence is not privacy sensitive. Some work considered collaborative intrusion detection rather than distributed IDS [1]. Different from distributed IDS, this paper focuses on privacy preservation and mitigates the presence of internal attacks, both by single and multiple colluding nodes. Finally adaptive aggregation algorithms have been pursued for mitigating attacks on FL [13]; yet they fail to cope with the hierarchical aggregation in unbalanced data for constrained EoT. To our knowledge such a defense mechanism has not been considered in the literature.

## III. System Model and Approach Overview

### A. System Model

FLIDS considers emerging IoT applications where computation is pushed to the network edge to alleviate demands on the communication infrastructure. Data collected by IoT devices is processed in part or fully by edge devices and then shared and/or aggregated by fog nodes. The paper focuses on the security of the involved edge devices by designing an effective IDS that suits the distributed computation framework while factoring in privacy concerns about sharing logs and alerts. The IDS fundamentally builds a machine learning model that uses collected intelligence, e.g., signatures of known attacks, and alerts from the network nodes in order to safeguard the devices and accurately classify unknown attacks. The underlying EoT architecture motivates the need for the IDS realization in a distributed manner, especially given the large access logs and the numerous devices that are part of the EoT system.

A distributed IDS implementation can be simply taken in the form of sharing alerts among the edge devices; however, such an approach suffers serious shortcomings and is not deemed practical. First, edge nodes often are not directly reachable to each other. Therefore, alerts and logs are to be disseminated over multi-hop paths within the EoT. which not only requires encryption and management of cryptographic keys, but also imposes excessive communication overhead, given the scale of an EoT. Second, such an approach lacks any protection against malicious actors where wrong intelligence data is provided by some devices to degrade the IDS. Third, sharing alerts/logs would raise privacy concerns since the owners of edge nodes may not want to publicize the undergoing activities.

### B. Attack Model

The quality of a network-based IDS heavily depends on the data used in forming the traffic classification model. Given the evolving nature of EoT, existing intelligence databases are primitive and a lot of the relevant data is being collected in real-time from the network nodes. Hence, the robustness of the IDS is highly influenced by the trustworthiness of the EoT participants. A malicious participant can degrade the IDS performance by data poisoning attacks. Such data poisoning can be even orchestrated through collusion among multiple participants. To assess the severity of the collusive attack on IDS, we have conducted a study using the NF-UNSW-NB15-v2 dataset [5], and involving ten participants. The data is evenly split among the participants. The machine learning model formation is centralized in the study, which mimics the scenario where the model is maintained at a cloud server and then sent to edge nodes. About a 40% drop in true positive classifications is observed when as little as two nodes collude. The results, as discussed in detail in Section V, illustrate the severity of the increasing number of collusive attackers on the effectiveness of an IDS. FLIDS mitigates the threat of such a collusive attack.

### C. Approach Overview

As mentioned above, an EoT intrusion detection system should: (i) achieve the integrity of the underlying traffic classification model, (iii) limit the overhead of communication between devices, and (iii) enable privacy preservation. To achieve these design goals, FLIDS exploits both fog computing and federated learning. Fog computing orchestrates the fusion of security logs from the individual edge nodes to update the IDS, and shares the upgraded IDS with edge devices. FL allows forming an aggregated model out of multiple data-driven intrusion detection models. FL is an ideal framework for the aggregation of participant models while preserving the privacy and enabling the converges to a distributed learning mechanism with close accuracy to that of a centralized implementation [14]. We further improve the traditional FL mechanism to enable individual and collusive insider attack detection and real-time mitigation. As articulated in Figure 1, edge devices train their local models to determine the local weights and loss values. This information is then sent to the aggregators, i.e., fog nodes. Aggregation can be performed on multi-tier inter-fog node topology. The aggregated model is then sent back to the devices in order to improve the model using the collected security alerts. FLIDS identifies individual and collusive data poisoning attempts and employs a robust defense mechanism.

## IV. Distributed Federated Learning based IDS

This section presents the proposed FLIDS approach. We first formulate the problem and then describe the solution in detail.

## A. Problem Formulation

Considering the set of EoT devices, Ψ, and the set of fog nodes, Φ, we opt to form a robust IDS that overcomes the presence of adversarial poisoning of security logs while preserving the privacy of the involved participants (edge devices). FLIDS employs federated learning to enable privacy-preserving distributed aggregation of stealthy models. The following summarizes the notation for the various players:

- *Participants*: $\Psi = \{\psi_1, \psi_2, \ldots, \psi_n\}$, set of edge devices; each collects data and forms a local IDS. It can be a cellphone or any device with some processing capabilities.
- *Fog nodes*: $\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_m\}$, a fog node has significant processing capacity; the fogs nodes are interconnected and also have communication links to the cloud server.
- *Cloud server*: A single entity with major computational power.

Each participant $\psi_i$ establishes a local model that can be described by a set of weights ($w_i^t$), derived from the data during the training phase. Every subset of the participating devices transmit their local models to a fog node ($\varphi_i$) that aggregates them and forwards to another fog node or a cloud server. Figure 1 depicts the underlying system architecture. Suppose that an edge device $\psi_i \in \Psi$ has a training dataset $D_i(x_i, y_i)$ where $D_i$ contains the feature values $x_i$ and the label $y_i$. For each data sample $d \in D_i$, i.e., a record in the security log, we have a loss $f_d(w) = f(w, x_d, y_d)$, which reflects the contribution of $d$ to the accuracy of the ML model. The objective of a centralized-implementation of FL is to use the multi-round aggregated $w_i^t$ in order to find the global optimized parameter vector $w^*$ that minimizes the global loss function $F(w)$ of the aggregated:

$$w^* = \operatorname*{argmin}_{w} F(w) \tag{1}$$

However, when distributed training is pursued, the objective of the optimization is to minimize the global loss $F(w^f)$, which can be expressed as the minimization of $[F(w^f) - F(w^*)]$, where $w^f$ represents the final weights after many iterative aggregation steps. In the first step, the central server, e.g., the cloud, broadcasts the global initial parameter vector $w^0$ to all devices for synchronization. The weights at node $\psi_i$ is initially $w_i = w^0$. The total (global) loss is computed as follows:

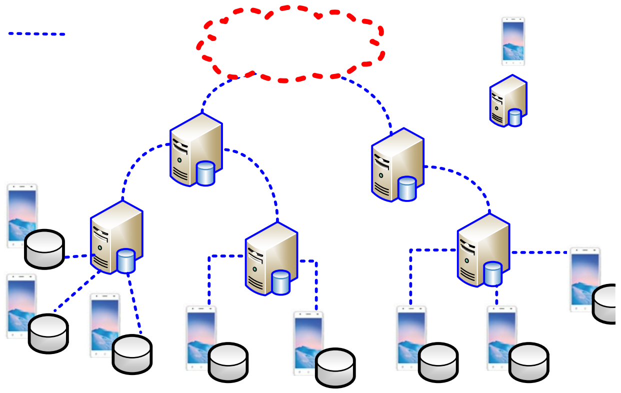

**Fig. 1.** The considered multi-tier network architecture. Security logs are processed at the edge of the network to form local models. The latter are forwarded to the cloud server through fog nodes that further aggregate the local models. The cloud server forms a global model out of the aggregated model and sends back the updated IDS to edge devices.

a) The local device $\psi_i \in \Psi$ updates its loss using:

$$F_i(w) = \sum_{d \in D_i} f_d(w_i) / |D_i| \tag{2}$$

where $|D_i|$ is the size of the dataset for $\psi_i$.

b) The weights are updated for each device using:

$$w_i(t+1) \leftarrow [w_i(t) - \eta \nabla F_i(w_i(t))] \tag{3}$$

where $\eta > 0$ is a constant denoting the learning rate, and $\nabla F_i(w_i)$ is the gradient of $F_i(w_i)$.

c) The server aggregates all device updates to improve the global model. In particular, the local weights will be averaged. The aggregation of the individual models will form the global model. Hence, the global loss function is:

$$F(w) = \sum_{\psi_i \in \Psi} |D_i| * F_i(w) / |D| \tag{4}$$

where $D = \bigcup_{\psi_i \in \Psi} D_i$.

d) The server then updates the global model and sends back the new weights to the devices, and so on.

In EoT, the aggregation pattern is hierarchical instead of being done at the cloud [15], where the weight update is defined by:

$$w_i(t+1) = \begin{cases} \sum_{i \in C^l} \frac{|D_i^l| [w_i(t) - \eta \nabla F_i(w_i)]}{|D^l|}, & if\ i\ is\ fog\ node \\ \sum_{i=1 \ldots N} \frac{|D_i^l| [w_i(t) - \eta \nabla F_i(w_i)]}{|D|}, & if\ i\ is\ cloud \end{cases} \tag{5}$$

Furthermore the weight $w_i(t)$ in a given round should be synchronized according to the aggregation. Although some edge nodes have sufficient training data, most participants will have relatively small amounts of data for training their local models. This could lead to unbalanced accuracy among local models and slows the convergence of the aggregation process. Weight updates will mitigate such unbalance. Thus, instead of receiving synchronized weights from the server, in EoT, the weights will be updated by intermediate fog nodes in the aggregation path. Therefore, we define the contribution of the participant based on their weight difference from the recently updated weights along the aggregation path of such a round.

As pointed out in Section III, the accuracy of local models could be maliciously degraded in hope to yield an inaccurate aggregated IDS. Such malicious behavior could be conducted by one or multiple colluding participants. To detect such an attack and identify the actors, we formulate the problem as follows: Given the participant weights contribution, and a fraction, $\alpha < 0.5$, of the participants is malicious (the remaining $1 - \alpha$ fraction are normal), we need to determine the subset of colluding nodes that diminish the IDS detection rate for any targeted class. The problem is combinatory of all possible grouping over participant nodes. In fact, it is in essence *k*-set partitioning problem, which is shown to be NP-hard. Hence, FLIDS pursues an iterative two-step heuristic. First, a metric is defined to detect the malicious nodes by monitoring the node's contribution to the model. Then the suspicious nodes are ranked and a collusive subset is determined by a successive reinforcement policy on the impact on the IDS accuracy.

## B. Attack Detection

The aim of FLIDS is not only the formation of an IDS for EoT but also the mitigation of any internal individual or collusive data poisoning attack. By training a local model using poisoned inputs, a malicious participant opts to mislead the system and yields an inaccurate global model. The local model in this case would be dissimilar to a model that is trained with a genuine dataset; such a dissimilarity and will be manifested with a very low detection rate of the targeted label (malicious behavior/alerts). In another word, the quality of the participant's models will depend on the used data. FLIDS exploits the divergence of the participant's models in order to detect such the incorporation of poisoned security logs. Although, the individual model can be evaluated based on their detection rate at the evaluation stage, the multi-layer aggregation prevents the identification of the colluding nodes since the aggregation is incremental along the path from the individual nodes to the cloud server and thus depends on the precedence relationship of node updates. Therefore, the effect of any poisoning attack will be propagated successively toward the server.

As explained earlier, the multi-layer aggregation is pursued due to the heterogeneity of participant's data and its impact on the convergence of the federation. On the other hand, applying centralized aggregation at the server will impose many rounds of the communication. Indeed, when multiple fogs servers perform the partial model aggregation, the model can be trained faster and better communication-computation trade-offs can be achieved [12]. To sustain the advantages of incremental aggregation along the route to the cloud server, FLIDS defines the contribution of a participant's model based on its weight difference from the recently updated weights along the aggregation path in the round. The use of weight difference is because it assesses how the data of a particular node shifts the weight vector independent of the aggregation scheme. Given our assumption that most nodes are benign, the divergence of a local model will be indication of suspicious behavior, i.e.,

$$\sum_{j\epsilon F} w_i^j(t+1) - \widehat{w}_A^j(t+1)\,, \forall \psi_i \in \Psi \tag{6}$$

where $\widehat{w}_i^j(t+1)$ is the $j^{th}$ entry in the weight vector of node $i$ in round $t+1$, $\widehat{w}_A$ reflects the aggregated model at the next fog node on the aggregation path towards the cloud, and $F$ is the set of model parameters.

The next issue to be addressed is how the detection will be conducted. The cloud in each round will receive the weight difference to be used for the aggregation, and use Eq. (1) in order to detect the possible list of attackers, i.e., by comparing the models provided by the various aggregation paths in each round. The server will inform the respective fog nodes which in turn apply the same process. FLIDS attack mitigation mechanism will be discussed next.

## C. *Defense mechanism*

The mechanism discussed above previously detects suspicious patterns without explicitly determining the set of attackers, i.e., malicious edge nodes. In other words, while the detection mechanism can infer the potential sources of inaccurate local models (poisoned data) and rank them, it cannot guarantee for sure the cardinality of the set of colluding nodes. To identity colluding attackers, the problem can be formulated using Q-learning. Q-learning is a reinforcement learning mechanism that learns the best action in a particular state without any assumption about the environment. In our case, the objective is to find the smallest fraction β of the malicious weight vector, where for a given a set of $N$ devices ranked according to the weight updates, as described in Eq. (1), we need to find a subset of $K$ nodes to eliminate in order to improve the aggregated model accuracy.

Reinforcement learning (RL) considers a set of states, $S$, and learns how to select actions per state, including state transitions. Executing an action in a specific state provides the agent with a reward; the goal is to maximize the agent's total reward. This effectively motivates the selection of a current action based on the potential future reward. In our case, the contribution vector can be normalized to the one with maximum contribution and considered as a state. An action is to identify the subset of nodes $k$ whose models are being aggregated. The reward is defined as the update in the detection rate of the target label and is computed after each action selection. However, such RL will require very long training time to come up with the best action for each state, and would not suit EoT. Therefore, we promote a greedy and deterministic approach that reduces Q-learning complexity using a three-state iterative algorithm (Fig. 2).

Our algorithm monitors the detection rate of the target class for τ rounds. The objective is to alleviate any uncertainty due to under-fitting in the primary round that may occur due to the global imbalance among classes in the security logs used for generating the local models. When FLIDS observes a consecutive decrease in true positives (detection rate) over rounds, i.e., $TP_t < TP_{t-1}$, or that the detection rate does not grow monotonically with the number of rounds, it switches to the discard state where it blacklists the node whose contribution deviates the most from the vectors of other nodes. The monotonic increase condition ensures the convergence to the highest possible detection rate. Given the maximum number of round (γ) and the overall sample size $N$ in a target class, FLIDS requires that in a round $c > \tau$, the detection rate will be at least $c * N/\gamma$. Upon blacklisting the selected node, FLIDS will probe for ß rounds to assess the reward for discarding a node from the aggregation process. Due to the distributed execution, a discarded edge device cannot make any inference about the exclusion of its local model and will continue to receive the latest version of the global model to upgrade its IDS. While being in the discarded state, the device's local model will continue to be assessed by the corresponding fog node, yet without being aggregated. Improvement in the model accuracy will be penalized; if the penalty grows consistently for the blacklisted node, FILDS switches to the probing (initial state) state. The assessment of the local model is reflected as scores where blacklisted nodes gets penalized if their exclusion improves the accuracy. On the other hand, accumulating insignificant penalty over γ rounds will eventually qualify the

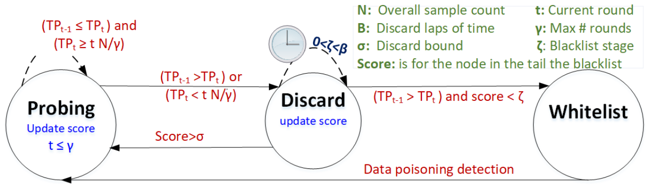

**Fig. 2.** The state transition diagram for FLIDS simplified Q-learning process

node for being whitelisted and transitioning to the normal state. However, the score is not cleared and frequent blacklisting/whitelisting of this particular node will be counted.

Being in the probing state does not qualify the blacklisted node's model for inclusion in the aggregation process. The state diagram in Fig. 2 reflects the status maintained for the global system. By excluding the most-deviating device from the global model, the list of devices becomes smaller in the next round and the attacker detectability increase for other malicious nodes; consequently, FLIDS progressively eliminates colluding devices. The defense mechanism will be applied in a distributed IoT network. Contrarily to existing work, we employ an edge-fog-cloud hierarchical topology. After every local update from an edge device, a fog (aggregator) node tracks the device's contribution that needs to be communicated to the cloud. The cloud runs FLIDS to maximize the long-term total reward; our algorithm can converge faster than a Q-learning mechanism due to the probing state that it is not randomized. After the learning step, the approach infers the potential decision output and will alleviate the impact of the individual or colluding attackers.

## V. Validation Experiments

To validate the effectiveness of FLIDS, we considered the recently published NF-UNSW-NB15-v2 dataset [5] that uses NetFlow-based features. Alerts are associated with network addresses; hence we have distributed the data according to the devices that triggers the alert, i.e., dispatch the alert to the participant according to the destination IP address. The dataset has 41 features. We performed RBF features selection and extracted the eight most relevant features. We mapped each attack label to eight one-hot encoded classes of begin and malicious behaviors. The samples for the dataset used for the training and validation is summarized in Table 1. We focused only on the ten nodes that report any of these attacks.

### A. Experiment Setup

We compare the performance of FLIDS to a centralized IDS. For such a centralized system, all security logs are consolidated and processed at the cloud, where we run a feedforward multilayer neural network architecture that contains two hidden layers using Rectified Linear Unit (RLU) as an activation with 64 neurons. The last layer of the network is a SoftMax layer with 8 output classes. We used cross entropy as the loss function. For FLIDS, we split the data records according to the IP destination addresses in order to make the simulation consistent with real setups. We have removed the source and

**Table 1:** The adversary success rate for various data precisions.

| Label | # Training Samples | # Test Samples |
|---|---|---|
| 1. Analysis | 596 | 1703 |
| 2. Backdoor | 511 | 1658 |
| 3. Benign | 573925 | 1721295 |
| 4. DoS | 1481 | 4313 |
| 5. Exploits | 7866 | 23685 |
| 6. Fuzzers | 5496 | 16814 |
| 7. Generic | 4102 | 12458 |
| 8. Reconnaissance | 3176 | 9603 |

the destination features so that they do not provide evidence of identifying attackers and allow the classifier to rely only on alerts in the security logs. The performance of the formed IDS is measured using the following metrics:

- The accuracy, F1 scores, recall and precision, to evaluate the performance of FLIDS and different ML models.
- The true positive, which reflects successful detection of a particular malicious behavior. Since the accuracy, F1, recall and precision factor in the true negative, they may not be sufficiently indicative when the true negatives dominate. Checking true positives will mitigate such an issue.

We vary the number of rounds for the federation, and the percentage of attackers among participants. The validation experiments and results are discussed next.

### B. Attack performance analysis

We first present the performance of FLIDS using the neural networks, Adaboost and Naives Bayes classifiers. Table 2 shows the performance of FLIDS for the individual classes in the absence of internal data poisoning attacks. FLIDS outperforms AdaBoost and Naives Bayes in terms of accuracy, precision, recall and F-score. The results demonstrate the effectiveness of the federation and neural networks. Fig. 3 illustrates the impact of the collusion on the attack detectability with 10 epochs of centralized training. The curves for "Defense" in the figure reflects the exclusion of the detected attackers based on FLIDS' strategy. We can see clearly that the detection rate is decreasing with the increase of the percentage of attackers among the edge nodes. In particular, the detection of anomalous behavior for the targeted class is not possible when the number of colluding attackers reaches 40%.

Fig. 4 illustrates the effectiveness of FLIDS in differentiating among malicious and well-behaving nodes. The number of misbehaving nodes (attackers) in Fig. 4(a) and 4(b), are two and three, respectively. The y-axis reflects the metric defined in Eq. 1. The attacker appears in bold legend in the

**Table 2:** Effect of dropout threshold on the adversary success rate.

| | AdaBoost | | | | Naives Bayes | | | | FLIDS | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Label | Accuracy | Precision | Recall | F-Score | Accuracy | Precision | Recall | F-Score | Accuracy | Precision | Recall | F-Score |
| #0 | 0.999 | 0 | 0 | 0 | 0.650 | 0 | 0.99 | 0.01 | 0.995 | 0.15 | 0.82 | 0.26 |
| #1 | 0.997 | 0.04 | 0.08 | 0.05 | 0.995 | 0.01 | 0.06 | 0.02 | 0.998 | 0.07 | 0.01 | 0.02 |
| #2 | 0.750 | 1 | 0.74 | 0.85 | 0.613 | 0.99 | 0.6 | 0.75 | 0.994 | 1 | 0.99 | 1 |
| #3 | 0.998 | 0.14 | 0 | 0.01 | 0.996 | 0.01 | 0.01 | 0.01 | 0.997 | 0.07 | 0.01 | 0.02 |
| #4 | 0.991 | 0.63 | 0.71 | 0.67 | 0.955 | 0.06 | 0.17 | 0.09 | 0.991 | 0.64 | 0.82 | 0.72 |
| #5 | 0.812 | 0.04 | 0.73 | 0.07 | 0.992 | 0.57 | 0.43 | 0.49 | 0.992 | 0.61 | 0.7 | 0.65 |
| #6 | 0.933 | 0.07 | 0.67 | 0.12 | 0.992 | 0.49 | 0.67 | 0.56 | 0.996 | 0.82 | 0.71 | 0.76 |
| #7 | 0.9937 | 0.45 | 0.82 | 0.58 | 0.993 | 0.44 | 0.85 | 0.58 | 0.997 | 0.73 | 0.7 | 0.72 |

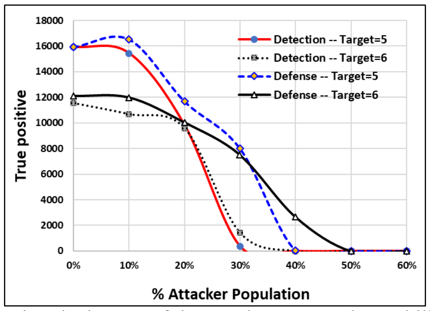

Fig. 3. Capturing the impact of the attack scope on detectability with and without FLIDS. The attack targets two specific classes, namely #5 and #6.

figure. We can see that an attacker that poisons the data to flip the IDS decision has the highest divergence factor. The detection rate is more visible when we have fewer colluding nodes. The results are also consistent for an increased number of attackers. As the number of collusive attackers grows to 3 (out of 10 nodes), FLIDS still performs reasonably well by detecting the attack; yet the difference becomes closer. This is explained by the increased collective influence of the attackers.

Fig. 5 reports the performance of FLIDS for a varying attacker population; here FL is employed and the model is trained for 55 rounds. When the percentage of malicious nodes is large, the number of true positives for the targeted class decreases significantly. The figure also captures the relationship between the number of collusive attackers and the defense mechanism success. Our defense mechanism alleviates the impact of the attack successfully, where overall FLIDS achieves 25% improvement. It is obvious that FLIDS cannot match the ideal result when none of the nodes is poisoning the security logs; this is mainly due to access to the aggregated local models rather than the raw data. Nonetheless, FLIDS still has positive impact on the detectability. The results are consistent with those of the centralized implementation in Fig. 3, and confirm the effectiveness of FLIDS.

## VI. Conclusions

This paper has presented FLIDS, a novel distributed and privacy-preserving IDS for EoT. FLIDS employs a federated learning with reinforcement mechanism in order to mitigate internal attacks that exploit data poisoning to degrade the IDS robustness. Our approach applies a hierarchical models aggregation mechanism that suits the constrained EoT setup. The aggregation process is further optimized to detect the bias imposed by individual and colluding malicious edge nodes. The detection methodology is based on the use of the quality of the model updates over time. We have proposed a lightweight defense mechanism that enforces the aggregation policy and mitigates collusive attacks. FLIDS is validated using a prominent dataset; the results have confirmed the effectiveness of FLIDS' detection methodology and defense algorithm.

## Reference

[1] E. Benkhelifa, T. Welsh and W. Hamouda, "A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Toward Universal and Resilient Systems," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3496-3509, Fourthquarter 2018.

[2] T. Sherasiya, H. Upadhyay, and H. B. Patel, "A survey: Intrusion detection system for internet of things," *International Journal of Computer Science and Engineering (IJCSE)*, 5(2), pp. 91–98, 2016.

[3] J. Liu, D. Yang, M. Lian and M. Li, "Research on Intrusion Detection Based on Particle Swarm Optimization in IoT," *IEEE Access*, vol. 9, pp. 38254-38268, 2021.

[4] A. Yang, X. Wang, Y. Sun, Y. Hu, Z. Shi and L. Sun, "Multi-Dimensional Data Fusion Intrusion Detection for Stealthy Attacks on Industrial Control Systems," *Proc. IEEE Global Comm. Conference (GLOBECOM)*, 2018.

[5] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "Towards a Standard Feature Set of NIDS Datasets," *arXiv:2101.11315v1*, 2021.

[6] H. Nkiama, S. Z. M. Said, and M. Saidu, "A subset feature elimination mechanism for intrusion detection system," *International Journal of Advanced Computer Science and Applications*, 7(4), pp 148–157, 2016.

[7] M. Eskandari, Z. H. Janjua, M. Vecchio and F. Antonelli, "Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices," *IEEE Internet of Things Journal*, 7(8), pp. 6882-6897, 2020.

[8] A. Yahyaoui, T. Abdellatif and R. Attia, "Hierarchical anomaly based intrusion detection and localization in IoT," *Proc. 15th Int'l Wireless Comm. & Mobile Computing Conf. (IWCMC)*, pp. 108-113, 2019.

[9] A. Tabassum, A. Erbad, A. Mohamed and M. Guizani, "Privacy-Preserving Distributed IDS Using Incremental Learning for IoT Health Systems," *IEEE Access*, 9, pp. 14271-14283, 2021.

[10]N. A. Al-Athba Al-Marri, B. S. Ciftler and M. M. Abdallah, "Federated Mimic Learning for Privacy Preserving Intrusion Detection," *Proc. IEEE Black Sea Conf. on Comm. & Net. (BlackSeaCom)*, Odessa, Ukraine, 2020.

[11]S.I. Shterenberg, and M.A. Poltavtseva, "A Distributed Intrusion Detection System with Protection from an Internal Intruder," *Aut. Control Comp*. Sci., Vol. 52, pp. 945–953, 2018.

[12]H. Al-Hamadi, I. -R. Chen, D. -C. Wang and M. Almashan, "Attack and Defense Strategies for Intrusion Detection in Autonomous Distributed IoT Systems," *IEEE Access*, vol. 8, pp. 168994-169009, 2020.

[13]Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. "Byzantine-robust distributed learning: Towards optimal statistical rates," arXiv preprint arXiv:1803.01498, 2018

[14]H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proc. Int'l Conf. on Artificial Intel. and Statistics (AISTATS)*, 2016.

[15]L. Liu, J. Zhang, S. H. Song and K. B. Letaief, "Client-Edge-Cloud Hierarchical Federated Learning," *Proc. IEEE International Conference on Communications (ICC)*, Dublin, Ireland, 2020.

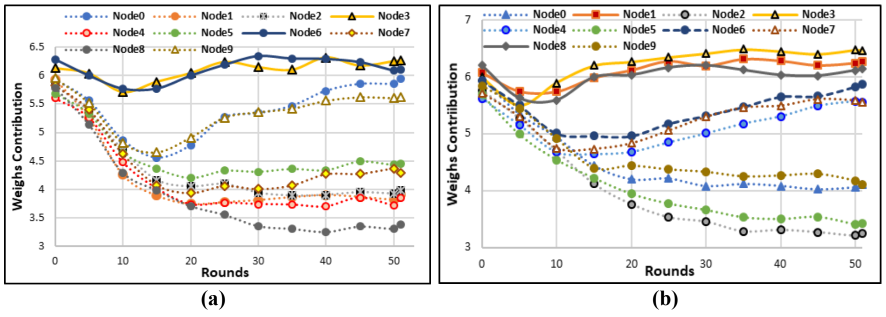

Fig. 4. Assessing FLIDS' attack detectability as collusion increases, where: (a) nodes 3 and 6 collude, and (b) nodes 1, 3, and 6 are involved.

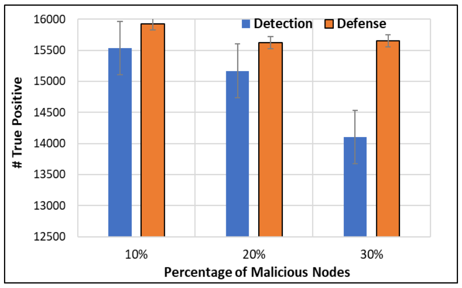

Fig 5. The effect of collusive attack when federated learning is employed with and without FLIDS defense mechanism.