


Article

# Robust Federated Learning for Mitigating Advanced Persistent Threats in Cyber-Physical Systems

Ehsan Hallaji <sup>1,\*</sup> , Roozbeh Razavi-Far <sup>1,2</sup>  and Mehrdad Saif <sup>1</sup> 

<sup>1</sup> Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada; roozbeh.razavi-far@unb.ca (R.R.-F.); msaif@uwindsor.ca (M.S.)

<sup>2</sup> Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada

\* Correspondence: hallaji@uwindsor.ca

**Abstract:** Malware triage is essential for the security of cyber-physical systems, particularly against Advanced Persistent Threats (APTs). Proper data for this task, however, are hard to come by, as organizations are often reluctant to share their network data due to security concerns. To tackle this issue, this paper presents a secure and distributed framework for the collaborative training of a global model for APT triage without compromising privacy. Using this framework, organizations can share knowledge of APTs without disclosing private data. Moreover, the proposed design employs robust aggregation protocols to safeguard the global model against potential adversaries. The proposed framework is evaluated using real-world data with 15 different APT mechanisms. To make the simulations more challenging, we assume that edge nodes have partial knowledge of APTs. The obtained results demonstrate that participants in the proposed framework can privately share their knowledge, resulting in a robust global model that accurately detects APTs with significant improvement across different model architectures. Under optimal conditions, the designed framework detects almost all APT scenarios with an accuracy of over 90 percent.

**Keywords:** federated learning; advanced persistent threats; robust aggregation; cyber security; malware triage



**Citation:** Hallaji, E.; Razavi-Far, R.; Saif, M. Robust Federated Learning for Mitigating Advanced Persistent Threats in Cyber-Physical Systems. *Appl. Sci.* **2024**, *14*, 8840. <https://doi.org/10.3390/app14198840>

Academic Editor: Fabrizio Marozzo

Received: 30 August 2024

Revised: 25 September 2024

Accepted: 27 September 2024

Published: 1 October 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The security of the network layer in cyber-physical systems (CPSs) is of paramount importance in various domains [1–3]. While conventional Intrusion Detection Systems (IDSs) are effective at flagging common cyber-attacks in network traffic, they often struggle to differentiate more advanced threats that use a sequence of different attack mechanisms to achieve malicious objectives [4,5]. This limitation prevents security experts from fully understanding the true objectives of the attacks once they are detected.

Advanced Persistent Threats (APTs) are a prime example of these sophisticated attacks that jeopardize network security in CPSs [4–6]. In contrast to commonly known cyber-attacks, APTs are continuous attacks that take place over time. These targeted attacks are designed to remain undetected in the system for the duration of the attack. Mitigating APTs requires more complex solutions, as conventional defense mechanisms often fall short in detecting these attacks [7]. Failure to flag APT attacks can lead to catastrophic consequences, including financial problems and loss of information. For example, in 2009, an APT attack named Stuxnet hacked into the Supervisory Control and Data Acquisition (SCADA) system and Programmable Logic Controllers (PLCs) of a nuclear plant, aiming at the physical destruction of this facility [8].

These challenges necessitate tailored solutions to eliminate APTs promptly. Many research efforts are dedicated to the detection of APTs using machine learning techniques [9,10]. Detection models are the core of malware triage, distinguishing sophisticated malware and prioritizing it over regular malware [11,12]. In this process, the characteristics of the

detected malware should be analyzed based on past events and the dynamics of the network so that the potential impact of the attack on the system can be estimated. This enables security teams to manage their resources and eliminate the threat more effectively. By doing so, organizations can identify and respond to APTs promptly, which, in turn, reduces the risk of long-term infiltration and damage [13]. In other words, utilizing this approach facilitates streamlining the response process, allowing security teams to focus on the most critical threats first, such as APTs, while efficiently handling less severe malware incidents.

Existing APT detection models can be categorized into anomaly detection and signature-based detection models [7]. Signature-based models usually employ supervised learning to distinguish between benign and malicious traffic [14,15]. Nonetheless, given the small proportion of APT samples in network traffic, signature-based models may fail during operation despite the high accuracy they exhibit during training (i.e., the accuracy paradox) [16]. Moreover, signature-based methods may struggle to detect unknown APT samples, as their supervised models usually recognize data distributions similar to those of their training data [17]. Conversely, anomaly detection methods are designed to detect abnormal patterns that may indicate an adversary in the network [10,14]. However, this approach usually creates an excessive workload for the security team when the false alarm rate increases. As a solution, rule mining can be used to rank the detected anomalies and assist in triaging them [10].

The similarity between the signature-based approach and the anomaly detection approach stems from their need for large amounts of training data [7,14]. Since these models depend on centralized data offerings, their scope of knowledge is quite limited, and hence the effectiveness of detection is adversely affected, especially against new APTs. Similar to other data-driven methods, building an automated system for the detection and triage of APTs requires ample high-quality data. Nevertheless, such data are hard to come by for several reasons. Firstly, due to the complexity and long-term nature of APTs, simulating these attacks in a lab setting is very challenging and time-consuming. Secondly, real-world data associated with real-world cases of APTs are very rare, as organizations are reluctant to share their network data publicly due to security concerns. One potential solution to this problem is using federated learning (FL), which enables collaborative learning via parameter sharing and data decentralization [18]. Although FL can be used as a solution for data decentralization in distributed training, its privacy and security are still concerning, as the FL surface itself may be subjected to adversaries, as investigated in the literature [19–21].

To address the aforementioned challenges, this paper makes the following contributions:

- To tackle the issue of data scarcity and secure knowledge sharing, we design a secure detection framework through which different organizations can jointly train a global model for malware triage without sharing their private data, thereby adhering to privacy regulations.
- In addition, adaptive clipping is used to safeguard the global model against adversaries that may degrade detection performance.
- To ensure the detection model does not experience the accuracy paradox, the detection framework is evaluated using highly imbalanced and non-i.i.d. datasets across different nodes.
- Experiments are conducted using public records associated with 15 APT cases to ensure the practicality of the designed framework for real-world applications.

The remainder of this paper is organized as follows. Section 2 states the targeted problem and briefly reviews the required background for this paper. Section 3 presents the proposed methodology. Section 4 illustrates and analyzes the experimental results. Finally, this paper concludes in Section 5.

## 2. Background

The contribution of this paper lies at the intersection of APT defense and FL. This section briefly reviews each of these topics to clarify the connection between these fields in

this work. Then, the data utilized for simulating a case study is introduced at the end of this section.

### 2.1. Advanced Persistent Threats

The integration of physical components, a network layer, and an application layer in modern CPSs makes them susceptible to cyber-attacks [22–24]. An APT is one of the most concerning types of these threats [25]. These sophisticated attacks have very specific objectives and are executed gradually, with a tendency to maintain long-term access to the system. Moreover, the attackers operating an APT often orchestrate the attack themselves rather than using automated tools and codes. They are skilled and well resourced, combining several tools and techniques to carry out each step of the APT. Furthermore, the continuous and persistent nature of APTs makes them hard to detect by security teams, as they cannot definitively categorize an attack as an APT due to a lack of information. Thus, the security team must promptly detect any APTs in the system so that enough resources and manpower can be allocated to eliminate such threats.

The detection of APT attacks is often performed as part of the malware triage process, where different adversaries are ranked based on their degree of severity to set priorities for the security team. This data-driven process, however, is heavily reliant on the availability of data and historical patterns associated with critical threats such as APTs [5,15]. Several research endeavors have presented solutions for detecting APTs [14,26–28]. For instance, a risk management approach can be used to analyze the system's expected state [29]. Multi-label learning has been shown to be effective at categorizing each malicious sample into different groups to provide extended insight into the data [30]. A dynamic quarantine and recovery scheme was designed in [6] to minimize the impact of APTs when they are present in a system. Nonetheless, despite the scarcity of data for effectively combating APTs, there are not many studies that address this problem. Privacy and security concerns associated with revealing network data prevent organizations from sharing such data. This paper tackles this issue by employing FL.

### 2.2. Federated Learning

FL is a machine learning paradigm that collaboratively trains a global model over several nodes or edge devices while preserving privacy [18,31,32]. User privacy in FL is ensured through data decentralization, as each edge device forbids access to the local data it contains [33–35]. Instead, nodes train a model on their local data and communicate the locally optimized parameters to an aggregator server. The aggregator receives these updates from edge devices and aggregates them into a global model [36]. Then, the parameters of this newly obtained model are sent to the edge devices so that they can update their local models to reflect the latest state of the global model. As this cycle continues, edge devices share the knowledge obtained from the data they contain and can improve their local models based on the information gathered from other nodes through parameter aggregation.

In the context of CPSs, FL can be used to build a global model to detect intrusions and cyber-attacks without the need to share private data [37–39]. Each organization or node of the network that has a local malware detection model can constantly improve its model based on the information that other nodes have acquired from their data, without having access to such data. However, employing FL creates an extra communication surface between the nodes of the network. As a result, if a node in this network becomes rogue, it can jeopardize the network through different adversaries [19,21,40–42]. This problem, however, is addressed by employing robust aggregation principles in our network, as explained in Section 3.

### 2.3. Case Study

To simulate real-world APTs in computer networks, we considered the data introduced in [11]. These data were created based on known APTs with publicly available reports. The

dataset contains 2086 APT samples obtained from 15 different APTs, as listed in Table 1. A total of 9021 non-APT malware samples are also included in the data. The objective of this case study is to distinguish between APT and non-APT attacks so that malware triage can be performed accordingly. Table 1 indicates the population of samples associated with each APT mechanism. To minimize latency, samples were recorded using static features that could be extracted with minimal delay. Initially, 4000 features were extracted using the PEFrame tool. These features consisted of optional headers, MS-DOS headers, file headers, obfuscated string statistics, Mutex, Packer, buckets, and imported APIs. As investigated in [12], fewer than 300 of these features carry the required information for the task at hand. As a result, we selected 264 of the originally extracted features. Feature selection was performed based on the importance score given by random forest, as explained in [12].

**Table 1.** APT mechanisms included in the dataset [11]. Class populations are indicated accordingly.

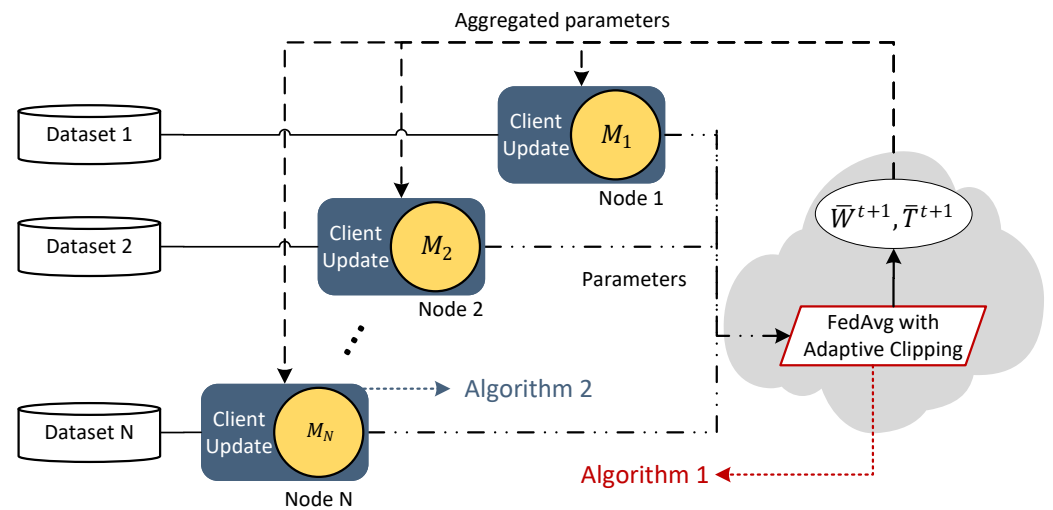
Label	Class	No. Samples
1	APT28	68
2	APT29	205
3	APT30	101
4	Carbanak	105
5	Desert Falcon	45
6	Hurricane Panda	315
7	Lazarus Group	58
8	Mirage	54
9	Patchwork	559
10	Sandwork	44
11	Shiqiang	31
12	Transparent Tribe	267
13	Violin Panda	23
14	Volatile Cedar	35
15	Winnti Group	176
16	Non-APT Malware	9021

### 3. Methodology

Figure 1 depicts the overall structure of the APT detection framework. Considering a set of nodes using a neural network model  $M$  in the FL network, each node contains a local dataset  $D = \{X, Y\}$ , where  $X = \{x_1, x_2, \dots, x_m\}$  and  $Y = \{y_1, y_2, \dots, y_m\}$  denote the set of samples and labels, respectively. We assume that  $X \in \mathbb{R}^n$  and  $Y \in \mathbb{N}$ . Furthermore, the data are non-i.i.d. across different nodes. In a security context,  $X$  denotes the samples recorded from network traffic, whereas  $Y$  denotes the labels specifying the attack mechanisms with which the recorded samples are associated.

At time step  $t$ , a model is trained at node  $k$  on  $D_k^t$  and optimizes the parameters, which in our case are the weights of the network  $W_k^t = \{w_1, w_2, \dots, w_l\}$ , where  $l$  is the number of layers.  $W_k^t$  is then communicated to the aggregation server. The server receives  $W_k^t$  for  $N$  nodes and commences the aggregation process. As a common standard, the server can zero out abnormally large values with a threshold  $T_0$  to eliminate the effect of potential data poisoning attacks. Assuming that  $w_{ij}$  indicates the  $j$ -th element of  $w_i \in W_k^t$ , then  $\forall w_{ij} \in W_k^t$ ,  $1 \leq k \leq N$ :

$$\text{Clip}(w_{ij}, T_0) = \begin{cases} w_{ij} & \text{if } |w_{ij}| \leq T_0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$



**Figure 1.** Block diagram of the APT detection framework. Algorithms 1 and 2 are used for robust aggregation and client updates, respectively.

If a client becomes rogue and acts maliciously, they can disrupt the global model by communicating a set of poisoned parameters. While such attacks may vary in their objectives, the adversarial task of a malicious client is generally to maximize the loss of the global model as follows:

$$\max_{\zeta_k} \mathcal{L}(\bar{W}_k^t, \zeta_k), \tag{2}$$

where  $\mathcal{L}$  is a loss function that returns the global model error given poisoned updates obtained by perturbation  $\zeta_k$ . Depending on the attack mechanism (e.g., model poisoning, data poisoning),  $\zeta_k$  can be applied to the local data as  $D_k^t + \zeta_k$  or directly to the resulting gradients as  $W_k^t + \zeta_k$ .

The server then uses the clipped weights to aggregate the updates. Given  $N$  nodes in the network, Federated Averaging (FedAvg) [18] is defined as follows:

$$\text{FedAvg}(\{W_k\}_{k=1}^N) = \frac{1}{N} \sum_{k=1}^N W_k^t \tag{3}$$

The server continuously receives  $W_k^t$  from nodes and computes  $\text{Clip}(W_k^t, T_0)$  to obtain the clipped weights  $\hat{\Delta}_k^t$ .  $\hat{\Delta}_k^t$  is then passed as  $\text{FedAvg}(\{\hat{\Delta}_k^t\}_{k=1}^N)$  to update the weights of the global model  $\bar{W}^{t+1}$  for the next iteration  $t + 1$  (see line 10 in Algorithm 1).

Nonetheless, finding a fixed threshold for  $T_0$  that optimally works for all nodes and datasets is not feasible in most cases. As a solution, one can dynamically adjust the threshold  $T$ . We follow the adaptive clipping approach proposed in [43] and employ an adaptive update rule.  $T$  is adjusted to find an approximate value at a specified quantile of the update norm distribution. Assuming that  $\gamma \in [0, 1]$  is the targeted quantile, for any  $T$ , the  $\text{Clip}(\cdot, \cdot)$  function can be reformulated to the loss function  $J$  with the following definition:

$$J(X, T) = \begin{cases} (1 - \gamma)(T - X) & \text{if } X \leq T \\ \gamma(X - T) & \text{otherwise} \end{cases} \tag{4}$$

The targeted quantile is selected based on the expected risk of adversaries in the system. Similar to most anomaly detection-based defense mechanisms in federated learning, adjusting this parameter involves a trade-off between security and performance. For large values of  $\gamma$ , gradients are aggressively clipped, which results in highly secure aggregation. Conversely, choosing small values of  $\gamma$  results in a more accurate model with a higher risk of being compromised.

---

**Algorithm 1:** FedAvg with Adaptive Clipping

---

**Input:** Local weights  $W_k^t$ , local thresholds  $T_k^t$   
**Output:** Aggregated weights  $\bar{W}^{t+1}$ , averaged threshold  $\bar{T}^{t+1}$

- 1 **for** each time step  $t$  **do**
- 2     **for** each node  $k, 1 \leq k \leq N$  **do**
- 3          $\Delta_k^t, T_k^t \leftarrow \text{ClientUpdate}(\bar{W}^t, \bar{T}^t)$
- 4     **end for**
- 5      $\bar{T}^{t+1} = \frac{1}{N} \sum_{k=1}^N T_k^t$
- 6     **for**  $\forall W_k^t, 1 \leq k \leq N$  **do**
- 7          $\hat{\Delta}_k^t = \text{Clip}(\Delta_k^t, \bar{T}^{t+1})$
- 8     **end for**
- 9      $\bar{\Delta}^t = \text{FedAvg}(\{\hat{\Delta}_k^t\}_{k=1}^N)$
- 10     $\bar{W}^{t+1} = \bar{W}^t + \eta_a \bar{\Delta}^t$
- 11 **end for**

---



---

**Algorithm 2:** Client Update

---

**Input:** Global weights  $\bar{W}^t$ , global threshold  $\bar{T}^t$ , dataset  $D_k^t$ , learning rate  $\eta$ ,  
threshold learning rate  $\eta_T$   
**Output:** Local weights  $W_k^t$ , local threshold  $T_k^t$

- 1  $W_k^t \leftarrow \bar{W}^t$
- 2  $T_k^t = \bar{T}^t$
- 3 **for** batch  $b \in X$  **do**
- 4      $W_k^t \leftarrow W_k^t - \eta \nabla J(W_k^t, b)$
- 5 **end for**
- 6  $\Delta_k^t \leftarrow W_k^t - \bar{W}^t$
- 7  $\Delta_k^t \leftarrow \Delta_k^t \cdot \min\left(1, \frac{T_k^t}{\|\Delta_k^t\|}\right)$
- 8 Calculate  $T_k^{t+1}$  using  $T_k^t, \eta_T$ , and (7)
- 9 **return**  $\Delta_k^t$  and  $T_k^{t+1}$

---

Aiming to find an update rule that leads to the optimal  $T$  for a specified value of  $\gamma$ , we treat the above formulation as an optimization problem. To do so, the average derivative of  $J$  for timestep  $t$  is calculated as follows:

$$\begin{aligned} \bar{J}'(X, T^t) &= \frac{1}{m} \sum_{i=1}^m \begin{cases} (1 - \gamma) & \text{if } x_i \leq T^t \\ \gamma & \text{if } x_i > T^t \end{cases} \\ &= \frac{1}{m} \left( (1 - \gamma) \sum_{i=1}^m \mathbb{I}_{x_i \leq T^t} - \gamma \sum_{i=1}^m \mathbb{I}_{x_i > T^t} \right) \triangleq \frac{1}{m} \sum_{i=1}^m \mathbb{I}_{x_i \leq T^t} - \gamma \end{aligned} \tag{5}$$

The update rule for estimating  $T^{t+1}$  can be obtained as follows:

$$T^{t+1} \leftarrow T^t - \eta_T \left( \frac{1}{m} \sum_{i=1}^m \mathbb{I}_{x_i \leq T^t} - \gamma \right), \tag{6}$$

where  $\eta_T$  is the learning rate used for the threshold estimation. Equation (5) enhances the model's performance and robustness by dynamically adapting the threshold to changes in the update distribution.

In the obtained update rule in (6), the value of  $T$  linearly changes with a maximum of  $\eta_T$  in each iteration. This can be problematic when the desired  $T$  is significantly larger or



smaller than the current value. To address this problem, a geometric form of this update rule is employed to speed up and improve the convergence of the function, as follows:

$$T^{t+1} \leftarrow T^t \cdot \exp\left(-\eta_T \left(\frac{1}{m} \sum_{i=1}^m \mathbb{I}_{x_i \leq T^t} - \gamma\right)\right) \tag{7}$$

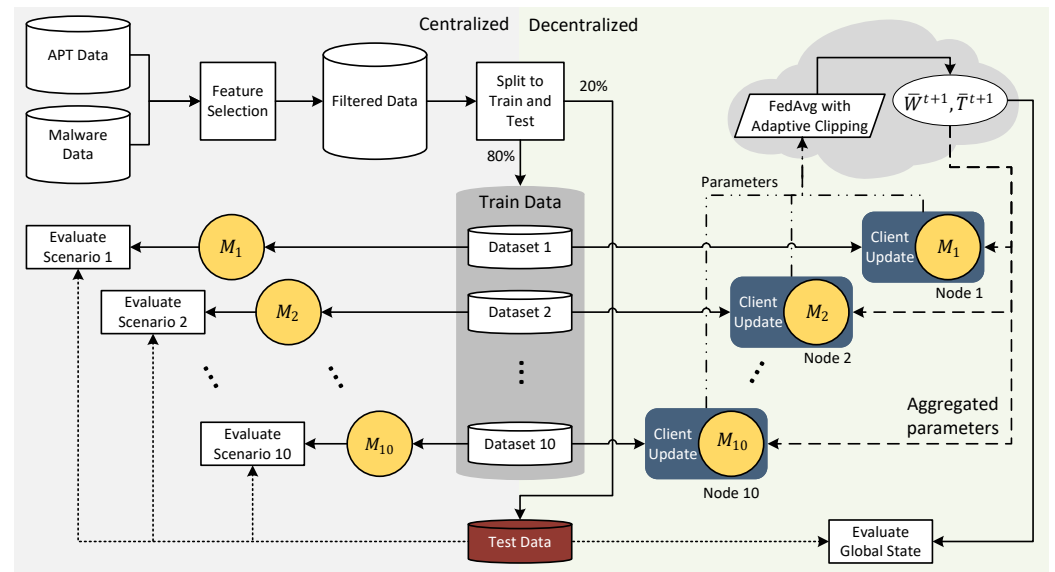
Each node communicates both  $W_k^t$  and  $T_k^t$  to the server. FedAvg with adaptive clipping is defined in Algorithm 1, which details the aggregation operation on the server side. Client-side operations are defined in Algorithm 2 based on the aforementioned formulations.

#### 4. Experimental Results

Figure 2 depicts the flow of experiments performed under centralized and decentralized settings, which are described as follows:

- Centralized training: Initially, an algorithm was selected to train ten different models on different subsets of training data. Each model was then evaluated on the test dataset to analyze its performance in a centralized setting.
- Decentralized training: In a decentralized setting, the designed framework was used to collaboratively train a model using ten different nodes that contained the same training subsets. Once the training was completed, the global state of the model, which was accessible by all nodes, was evaluated on the test dataset.

Each experiment was repeated ten times to ensure statistical reliability; consequently, the results were averaged to report the overall performance.



**Figure 2.** Block diagram of the experimental flow. Centralized and decentralized experiments are shown on the left and right sides of the diagram, respectively. The experimental dataset, consisting of APT and non-APT samples, was divided into training and test subsets to ensure that both centralized and decentralized experiments used the same datasets.

##### 4.1. Experimental Settings

Three neural network models—the One-Dimensional Convolutional Neural Network (1D CNN) [44], Denoising Autoencoder (DAE) [45], and Multi-Layer Perceptron (MLP)—were used as the base learners. All models used the Adam optimizer [46]. These techniques and the proposed FL framework were implemented in TensorFlow using Python 3.10.14. The FL framework was executed for 100 training rounds. The learning rate for adaptive clipping was empirically set to  $\eta_T = 0.2$ . The parameter settings for the local models are reported in Table 2.

To create non-i.i.d. datasets for different nodes of the FL network, ten datasets were considered, as shown in Table 3. All datasets contained non-APT malware along with five other APT classes that were randomly selected, with the condition that each APT class was included in at least one of the datasets. A test subset was created by selecting 20 percent of the samples from each class, as shown in Figure 2. The training datasets listed in Table 3 were drawn from the remaining 80 percent.

**Table 2.** Parameter settings used for edge local models.

Parameter	1D CNN	DAE	MLP
Epochs	500	100	10
Batch size	64	256	32
No. hidden layers	2	2	1
No. neurons	64, 64	128, 256	128, 64
Learning rate	0.001	0.001	0.002
Noise factor	-	0.5	-

**Table 3.** Scenarios used to create training datasets for each experiment. Each FL node uses only one of these datasets.

Dataset	Classes	No. Samples
Dataset 1	Malware, Transparent Tribe, Winnti Group, Carbanak, Desert Falcon, Violin Panda	7710
Dataset 2	Malware, APT29, Lazarus Group, Mirage, Sandwork, Violin Panda	7523
Dataset 3	Malware, Patchwork, Transparent Tribe, Carbanak, APT28, Volatile Cedar	8043
Dataset 4	Malware, Carbanak, APT28, Mirage, Shiqiang, Violin Panda	7441
Dataset 5	Malware, Winnti Group, Carbanak, APT30, Volatile Cedar, Violin Panda	7569
Dataset 6	Malware, Patchwork, Winnti Group, Lazarus Group, Mirage, Volatile Cedar	7921
Dataset 7	Malware, Hurricane Panda, Transparent Tribe, APT29, APT30, Shiqiang	7952
Dataset 8	Malware, Patchwork, Transparent Tribe, APT29, APT28, Lazarus Group	8141
Dataset 9	Malware, Transparent Tribe, Winnti Group, Mirage, Shiqiang, Violin Panda	7658
Dataset 10	Malware, Carbanak, APT30, Lazarus Group, Mirage, Desert Falcon	7506

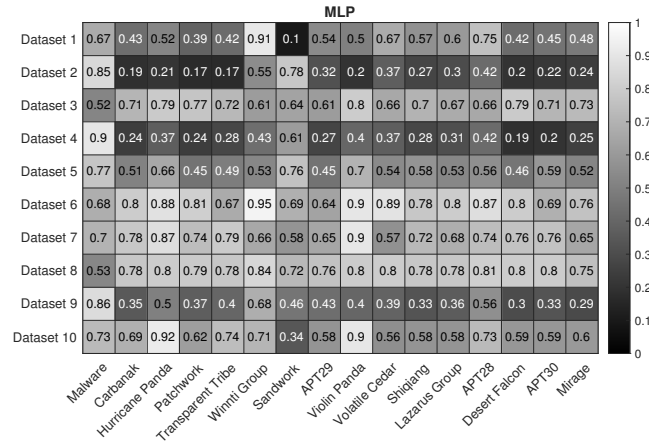
#### 4.2. Results

In order to study the effect of FL on constructing an accurate global model for APT detection, we first analyzed the results for the MLP, CNN, and 1D CNN in ten different scenarios. In each scenario, a model was trained on only one of the datasets listed in Table 3. The trained model was then evaluated on the test subset, which contained 20 percent of the samples from each class.

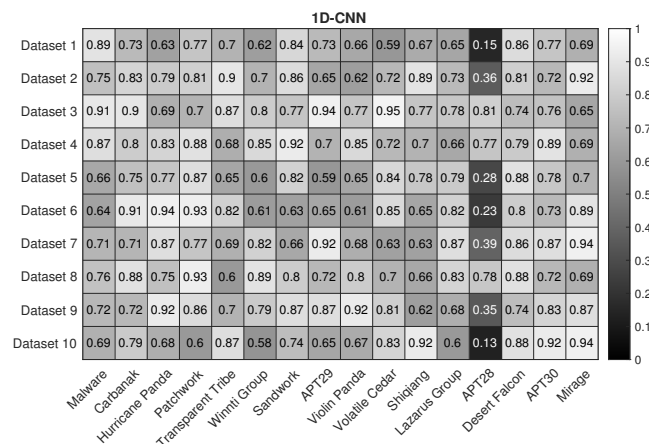
Figure 3 shows the obtained results using a heatmap for each algorithm. The datasets used for training the model in each experiment are indicated on the vertical axis, whereas the APT classes are specified along the horizontal axis. The color spectrum on the right side of each box indicates the range of colors used to differentiate between smaller and larger values. Figure 3a illustrates that the MLP was not able to detect APT classes with acceptable accuracy when trained on a subset of data with only a few APT classes included. Given that the majority of samples were non-APT malware, the model exhibited better performance in detecting malware. Among the datasets used for training, datasets six to eight resulted in better overall performance across all classes. Comparing the spectrum of colors in Figure 3b with the other panels in this figure, it can be concluded that the 1D CNN generally outperformed the MLP and DAE when trained on partial data. Nevertheless, this model seemed to exhibit poorer performance in detecting samples of the APT28 class than its competitors.



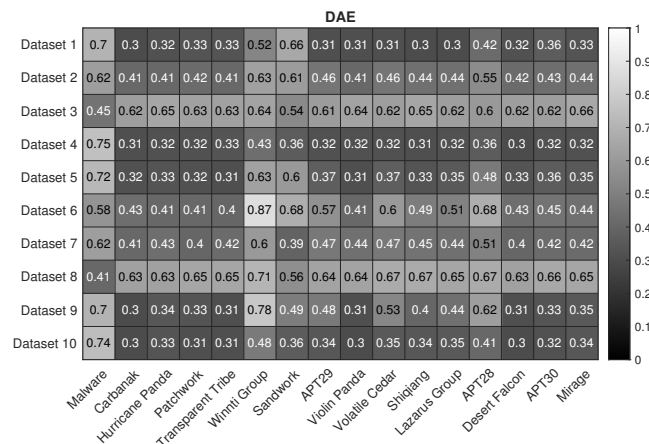
Interestingly, the 1D CNN’s performance was more consistent across different experiments and scenarios. On the other hand, the DAE exhibited poorer detection performance than the MLP and 1D CNN, as the color spectrum shown in Figure 3c is generally darker than those in Figure 3a,b. Regardless, the achieved results showed less variance than those of the MLP. Although using a more advanced structure with more hidden layers could probably boost the performance of the MLP, 1D CNN, and DAE, we relied on simple structures to study the effect of using FL on boosting detection performance.



(a)



(b)



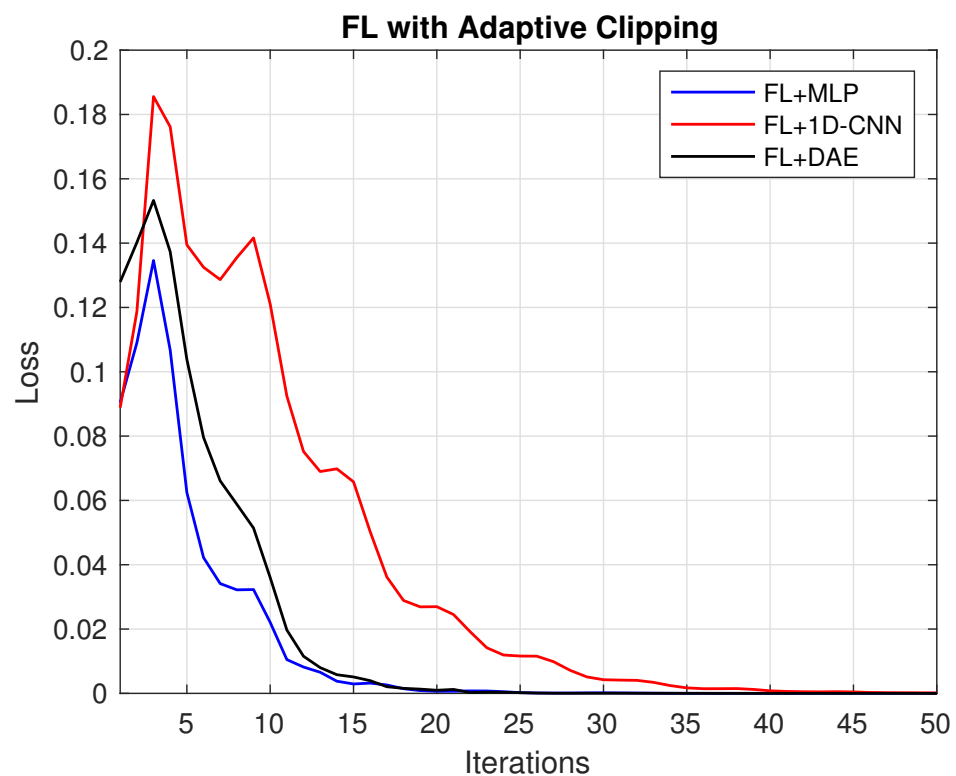
(c)

**Figure 3.** Success rates of detecting different APT classes without using FL. In each experiment, the indicated model is trained on only one dataset and evaluated on the test subset. (a–c) show the results of MLP, 1D-CNN, and DAE, respectively.

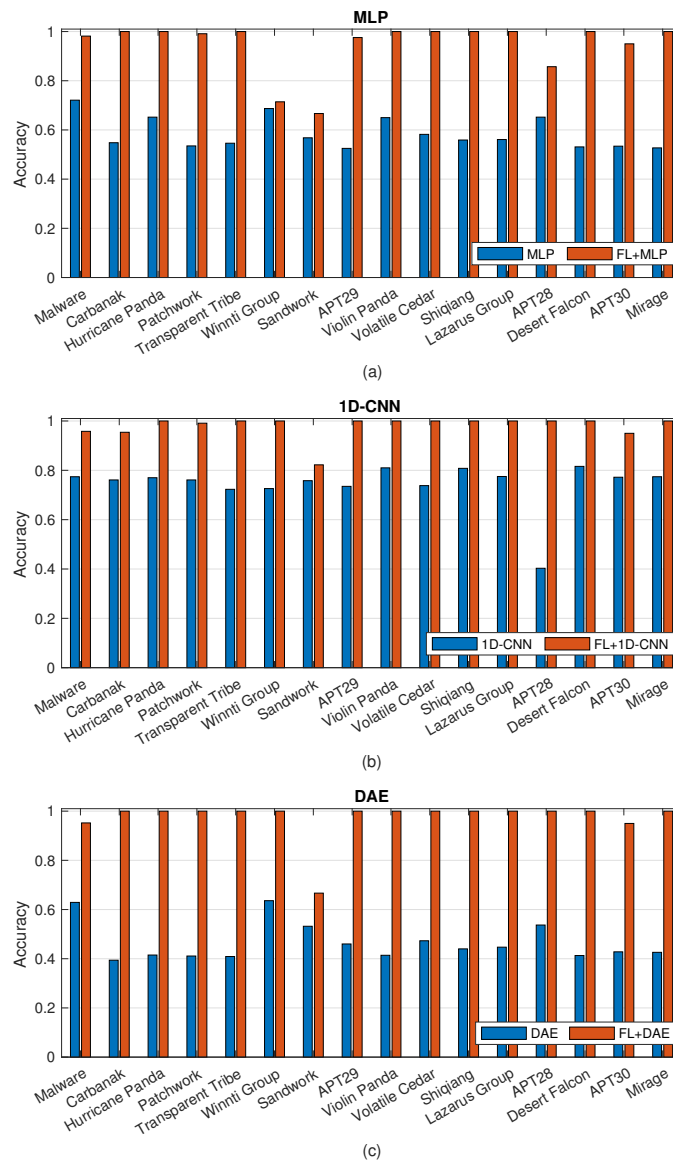
By integrating the selected classifiers into the designed FL structure, different nodes of the network can collaboratively train a global detection model to detect APT samples. Here, the FL network used ten nodes, where each node trained a local model on only one of the datasets listed in Table 3. Moreover, all of these nodes used a common network structure. Figure 4 depicts the cross-entropy loss of the global model through the iterations of the collaborative training. As nodes shared their parameters and updated their local weights in each round, the global loss decreased on the training data. This ensured that the global model was a perfect fit that successfully captured the APT distributions of each node. Furthermore, the global model converged rapidly and reached the optimal loss in fewer than 50 rounds of training. This implies the practicality of using FL in the collaborative training of a global APT detection model in CPSs. This application is vital to more data-sensitive CPSs, such as power systems and industrial control systems, where sensitive operational parameters and communication protocols can be targeted by APTs.

The obtained results shown in Figure 5 indicate a significant improvement in the detection performance of both the APT classes and malware. Regardless of the choice of model architecture at the edge of the network, the proposed FL framework outperformed the stand-alone models. While all three models exhibited outstanding results in detecting the APT classes, the 1D CNN outperformed the DAE and MLP combinations with the designed framework, as depicted in Figure 5b. The only class whose detection accuracy fell below 90 percent was Sandwork. Nevertheless, this pattern can also be seen for both the MLP and DAE in Figure 5a,c.

The observed improvements in the previous analysis indicate the significance of FL application across different CPSs. To begin with, accurate APT detection can prevent significant operational disruptions in industrial control systems. Another notable application of FL is in smart grids, which rely on distributed communication and control. These systems are often vulnerable to malicious activities and can benefit from the timely and accurate identification of adversaries.



**Figure 4.** Cross-entropy loss of training a global model using FL with adaptive clipping. Ten nodes participate in the FL training process.



**Figure 5.** Detection performance under centralized and decentralized settings. (a–c) show the results of MLP, 1D-CNN, and DAE, respectively.

### 5. Conclusions

Aiming to mitigate APT attacks in CPSs, this work designed a framework for the collaborative training of detection models used for the task of malware triage. Moreover, the proposed method tackles the issue of data scarcity for the task at hand by replacing data sharing with parameter sharing, thereby eliminating the privacy and security concerns associated with sharing private data between organizations. While the proposed method is based on FL, it also employs adaptive norm clipping to mitigate potential adversaries that may target the federated network during the collaborative training session. This adds an extra layer of security to the FL structure and safeguards the system against a wide range of adversaries. The proposed method was evaluated on real-world data for APT triage through several scenarios. Finally, the obtained results showed that the designed framework effectively builds a global model that successfully detects all 15 APT classes and the non-APT malware considered in this work. For future work, extending the design to employ privacy-preserving mechanisms can further enhance the model’s robustness. In addition, more APT variants can be considered in the simulations to evaluate the framework’s adaptability and generalization to new threats.

**Author Contributions:** Conceptualization, E.H. and R.R.-F.; methodology, E.H. and R.R.-F.; software, E.H.; validation, E.H.; formal analysis, E.H.; investigation, E.H.; resources, R.R.-F. and M.S.; data curation, E.H.; writing—original draft preparation, E.H.; writing—review and editing, R.R.-F. and M.S.; visualization, E.H.; supervision, R.R.-F. and M.S.; project administration, R.R.-F. and M.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under funding reference numbers CGSD3-569341-2022 and RGPIN-2021-02968.

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

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Publicly available datasets were analyzed in this study. This data can be found here: [https://github.com/GiuseppeLaurenza/L\\_F\\_Identifier/blob/master/dataset.tar.gz](https://github.com/GiuseppeLaurenza/L_F_Identifier/blob/master/dataset.tar.gz) (accessed on 26 September 2024).

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Kim, S.; Park, K.J.; Lu, C. A Survey on Network Security for Cyber-Physical Systems: From Threats to Resilient Design. *IEEE Commun. Surv. Tutor.* **2022**, *24*, 1534–1573. [CrossRef]
2. Ahmed Jamal, A.; Mustafa Majid, A.A.; Konev, A.; Kosachenko, T.; Shelupanov, A. A review on security analysis of cyber physical systems using Machine learning. *Mater. Today Proc.* **2023**, *80*, 2302–2306. [CrossRef]
3. Humayed, A.; Lin, J.; Li, F.; Luo, B. Cyber-Physical Systems Security—A Survey. *IEEE Internet Things J.* **2017**, *4*, 1802–1831. [CrossRef]
4. Huang, L.; Zhu, Q. A dynamic games approach to proactive defense strategies against Advanced Persistent Threats in cyber-physical systems. *Comput. Secur.* **2020**, *89*, 101660. [CrossRef]
5. Rahman, Z.; Yi, X.; Khalil, I. Blockchain-Based AI-Enabled Industry 4.0 CPS Protection Against Advanced Persistent Threat. *IEEE Internet Things J.* **2023**, *10*, 6769–6778. [CrossRef]
6. Yang, L.X.; Li, P.; Yang, X.; Xiang, Y.; Jiang, F.; Zhou, W. Effective Quarantine and Recovery Scheme Against Advanced Persistent Threat. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 5977–5991. [CrossRef]
7. Alshamrani, A.; Myneni, S.; Chowdhary, A.; Huang, D. A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 1851–1877. [CrossRef]
8. Langner, R. Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Secur. Priv.* **2011**, *9*, 49–51. [CrossRef]
9. Jia, Z.; Xiong, Y.; Nan, Y.; Zhang, Y.; Zhao, J.; Wen, M. MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning. In Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, 14–16 August 2024; pp. 5197–5214.
10. Dong, F.; Wang, L.; Nie, X.; Shao, F.; Wang, H.; Li, D.; Luo, X.; Xiao, X. DISTDET: A Cost-Effective Distributed Cyber Threat Detection System. In Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, USA, 9–11 August 2023; pp. 6575–6592.
11. Laurenza, G.; Aniello, L.; Lazzarotti, R.; Baldoni, R. Malware Triage Based on Static Features and Public APT Reports. In Proceedings of the Cyber Security Cryptography and Machine Learning, Cham, Switzerland, 29–30 June 2017; pp. 288–305.
12. Laurenza, G.; Lazzarotti, R.; Mazzotti, L. Malware Triage for Early Identification of Advanced Persistent Threat Activities. *Digit. Threat.* **2020**, *1*, 16. [CrossRef]
13. Sharma, A.; Gupta, B.B.; Singh, A.K.; Saraswat, V.K. Advanced Persistent Threats (APT): evolution, anatomy, attribution and countermeasures. *J. Ambient. Intell. Humaniz. Comput.* **2023**, *14*, 9355–9381. [CrossRef]
14. Abu Talib, M.; Nasir, Q.; Bou Nassif, A.; Mokhamed, T.; Ahmed, N.; Mahfood, B. APT beaconing detection: A systematic review. *Comput. Secur.* **2022**, *122*, 102875. [CrossRef]
15. Akbar, K.A.; Wang, Y.; Ayoade, G.; Gao, Y.; Singhal, A.; Khan, L.; Thuraisingham, B.; Jee, K. Advanced Persistent Threat Detection Using Data Provenance and Metric Learning. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 3957–3969. [CrossRef]
16. Thomas, C.; Balakrishnan, N. Improvement in minority attack detection with skewness in network traffic. In Proceedings of the Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, Orlando, FL, USA, 17–18 March 2008; Volume 6973, pp. 226–237.
17. Chen, P.; Desmet, L.; Huygens, C. A Study on Advanced Persistent Threats. In Proceedings of the Communications and Multimedia Security, Aveiro, Portugal, 25–26 September 2014; pp. 63–72.
18. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Ft. Lauderdale, FL, USA, 20–22 April 2017; Volume 54, pp. 1273–1282.

19. Hallaji, E.; Razavi-Far, R.; Saif, M.; Wang, B.; Yang, Q. Decentralized Federated Learning: A Survey on Security and Privacy. *IEEE Trans. Big Data* **2024**, *10*, 194–213. [[CrossRef](#)]
20. Bhagoji, A.N.; Chakraborty, S.; Mittal, P.; Calo, S. Analyzing Federated Learning through an Adversarial Lens. In Proceedings of the 36th International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; Volume 97, pp. 634–643.
21. Hallaji, E.; Razavi-Far, R.; Saif, M. *Federated and Transfer Learning: A Survey on Adversaries and Defense Mechanisms*; Springer: Berlin, Germany, 2022; pp. 29–55.
22. Han, S.; Xie, M.; Chen, H.H.; Ling, Y. Intrusion Detection in Cyber-Physical Systems: Techniques and Challenges. *IEEE Syst. J.* **2014**, *8*, 1052–1062.
23. Farajzadeh-Zanjani, M.; Hallaji, E.; Razavi-Far, R.; Saif, M. Generative-Adversarial Class-Imbalance Learning for Classifying Cyber-Attacks and Faults—A Cyber-Physical Power System. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 4068–4081. [[CrossRef](#)]
24. Zhu, T.; Ye, D.; Cheng, Z.; Zhou, W.; Yu, P.S. Learning Games for Defending Advanced Persistent Threats in Cyber Systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2023**, *53*, 2410–2422. [[CrossRef](#)]
25. Kayan, H.; Nunes, M.; Rana, O.; Burnap, P.; Perera, C. Cybersecurity of Industrial Cyber-Physical Systems: A Review. *ACM Comput. Surv.* **2022**, *54*, 1–35. [[CrossRef](#)]
26. Zhu, T.; Yu, J.; Xiong, C.; Cheng, W.; Yuan, Q.; Ying, J.; Chen, T.; Zhang, J.; Lv, M.; Chen, Y.; et al. APTSHIELD: A Stable, Efficient and Real-Time APT Detection System for Linux Hosts. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 5247–5264. [[CrossRef](#)]
27. Liang, H.; Li, C.; Li, X.; Jiang, S. APT Malware Classification Method Based on Feature Fusion. In Proceedings of the International Conference on Computer Information Science and Artificial Intelligence (CISAI), Kunming, China, 17–19 September 2021; pp. 456–462.
28. Pitolli, G.; Laurenza, G.; Aniello, L.; Querzoni, L.; Baldoni, R. MalFamAware: Automatic family identification and malware classification through online clustering. *Int. J. Inf. Secur.* **2021**, *20*, 371–386. [[CrossRef](#)]
29. Yang, L.X.; Li, P.; Yang, X.; Tang, Y.Y. A Risk Management Approach to Defending Against the Advanced Persistent Threat. *IEEE Trans. Dependable Secur. Comput.* **2020**, *17*, 1163–1172. [[CrossRef](#)]
30. Hallaji, E.; Razavi-Far, R.; Saif, M. Expanding analytical capabilities in intrusion detection through ensemble-based multi-label classification. *Comput. Secur.* **2024**, *139*, 103730. [[CrossRef](#)]
31. Razavi-Far, R.; Wang, B.; Taylor, M.E.; Yang, Q. An Introduction to Federated and Transfer Learning. In *Federated and Transfer Learning*; Springer International Publishing: Cham, Switzerland, 2023; pp. 1–6.
32. Zhang, C.; Xie, Y.; Bai, H.; Yu, B.; Li, W.; Gao, Y. A survey on federated learning. *Knowl.-Based Syst.* **2021**, *216*, 106775. [[CrossRef](#)]
33. Preuveneers, D.; Rimmer, V.; Tsingenopoulos, I.; Spooren, J.; Joosen, W.; Ilie-Zudor, E. Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study. *Appl. Sci.* **2018**, *8*, 2663. [[CrossRef](#)]
34. Asad, M.; Moustafa, A.; Ito, T. FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning. *Appl. Sci.* **2020**, *10*, 2864. [[CrossRef](#)]
35. Mothukuri, V.; Parizi, R.M.; Pouriyeh, S.; Huang, Y.; Dehghantanha, A.; Srivastava, G. A survey on security and privacy of federated learning. *Future Gener. Comput. Syst.* **2021**, *115*, 619–640. [[CrossRef](#)]
36. Qi, P.; Chiaro, D.; Guzzo, A.; Ianni, M.; Fortino, G.; Piccialli, F. Model aggregation techniques in federated learning: A comprehensive survey. *Future Gener. Comput. Syst.* **2024**, *150*, 272–293. [[CrossRef](#)]
37. Agrawal, S.; Sarker, S.; Aouedi, O.; Yenduri, G.; Piamrat, K.; Alazab, M.; Bhattacharya, S.; Maddikunta, P.K.R.; Gadekallu, T.R. Federated Learning for intrusion detection system: Concepts, challenges and future directions. *Comput. Commun.* **2022**, *195*, 346–361. [[CrossRef](#)]
38. Hallaji, E.; Razavi-Far, R.; Saif, M.; Herrera-Viedma, E. Label noise analysis meets adversarial training: A defense against label poisoning in federated learning. *Knowl.-Based Syst.* **2023**, *266*, 110384. [[CrossRef](#)]
39. Rahman, S.A.; Tout, H.; Talhi, C.; Mourad, A. Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning? *IEEE Netw.* **2020**, *34*, 310–317. [[CrossRef](#)]
40. Blanchard, P.; El Mhamdi, E.M.; Guerraoui, R.; Stainer, J. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
41. Fang, M.; Cao, X.; Jia, J.; Gong, N. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In Proceedings of the 29th USENIX Security Symposium. USENIX Association, Berkeley, CA, USA, 12–14 August 2020; pp. 1605–1622.
42. Reisizadeh, A.; Farnia, F.; Pedarsani, R.; Jadbabaie, A. Robust Federated Learning: The Case of Affine Distribution Shifts. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–12 December 2020; Curran Associates, Inc.: New York, NY, USA, 2020; Volume 33, pp. 21554–21565.
43. Andrew, G.; Thakkar, O.; McMahan, B.; Ramaswamy, S. Differentially Private Learning with Adaptive Clipping. In Proceedings of the International Conference on Neural Information Processing Systems, Online, 6–14 December 2021; Volume 34, pp. 17455–17466.
44. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [[CrossRef](#)]

- 
45. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.A. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 1096–1103.
  46. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.