algorithms

MDPI

Review

Comparative Review of the Intrusion Detection Systems Based on Federated Learning: Advantages and Open Challenges

Elena Fedorchenko [1,*,†], Evgenia Novikova [1,*,†] and Anton Shulepov [1,2]

[1] Saint Petersburg Institute for Informatics and Automation, Federal Research Center of the Russian Academy of Sciences, 199178 Saint Petersburg, Russia; ao-shyleo@yandex.ru
[2] Faculty of Computer Science and Technology, Saint Petersburg Electrotechnical University "LETI", 197376 Saint Petersburg, Russia
* Correspondence: doynikova@comsec.spb.ru (E.F.); novikova@comsec.spb.ru (E.N.)
† These authors contributed equally to this work.

Abstract: In order to provide an accurate and timely response to different types of the attacks, intrusion and anomaly detection systems collect and analyze a lot of data that may include personal and other sensitive data. These systems could be considered a source of privacy-aware risks. Application of the federated learning paradigm for training attack and anomaly detection models may significantly decrease such risks as the data generated locally are not transferred to any party, and training is performed mainly locally on data sources. Another benefit of the usage of federated learning for intrusion detection is its ability to support collaboration between entities that could not share their dataset for confidential or other reasons. While this approach is able to overcome the aforementioned challenges it is rather new and not well-researched. The challenges and research questions appear while using it to implement analytical systems. In this paper, the authors review existing solutions for intrusion and anomaly detection based on the federated learning, and study their advantages as well as open challenges still facing them. The paper analyzes the architecture of the proposed intrusion detection systems and the approaches used to model data partition across the clients. The paper ends with discussion and formulation of the open challenges.

Keywords: artificial intelligence; data partition; federated learning; Internet of Things; intrusion detection; machine learning; system architecture

check for updates

Citation: Fedorchenko, E.; Novikova, E.; Shulepov, A. Comparative Review of the Intrusion Detection Systems Based on Federated Learning: Advantages and Open Challenges. *Algorithms* **2022**, *15*, 247. https://doi.org/10.3390/a15070247

Academic Editors: Theodore B. Trafalis and Frank Werner

Received: 31 May 2022
Accepted: 6 July 2022
Published: 15 July 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 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

In order to provide an efficient response in a timely manner, the intrusion and anomaly detection systems collect and analyze a lot of data. Training an efficient analysis model for anomaly or attack detection also requires large volumes of data. These data often include confidential or sensitive data such as IP addresses, unique device or application identifiers, and location data that constitute personal data. Thus, there is a trade-off between data privacy and the security of the data subject. The paradigm of federated learning (FL) addresses this challenge and proposes a practical solution to build distributed intelligent systems in a privacy-preserving manner [1,2]. Moreover, recent research has shown that the analysis models trained in federated mode show comparable efficiency in the attack and anomaly detection [3–5]. Nevertheless, the application of the FL paradigm faces a set of important practical and theoretical design challenges. The key issues are as follows [1,6]:

- Edge device heterogeneity that may result in different data formats and attributes, and may require customizing local models.
- Non-IID (identically and independently distributed) data. It is a natural case that clients could have different features and/or label distribution as the data sources may be located in different geographical locations or time zones. The non-IID data also correspond to the cases with concept shift and/or drift, and skew in data amount held by different clients.

- Bias introduced by data owners' behavior, as the availability of the devices may change during the training process, and clients with more stable behavior may have a stronger impact on the results of training.
- FL system parameters tuning such as resource use on clients, learning throughput (number of clients, model complexity, etc).

The latter challenge tightly relates to the problem of the available datasets that could be used to train models and evaluate the parameters of the FL systems.

FL has recently become a highly researched problem, and there are now numerous studies in the scientific literature devoted to various theoretical and applied FL aspects. For example, in [7–9], the authors study the problem of the local model aggregation to ensure its robustness to the non-IID distribution of data across clients. Li et al. [10] address this problem by applying techniques of contrasting learning that utilize embeddings of the feature or model space to some intermediate representation space [11]. The challenges of constructing communication-efficient FL protocols in the context of resource constraint clients are discussed in [12–14]. For example, Zhang et al. [14] focus on accelerating computation and reducing communication overhead in FL frameworks secured by homomorphic encryption.

Other security and privacy aspects of the FL are studied in [15–17].

There are numerous studies that analyze the applicability of FL to different practical tasks. By now, the researchers have proposed various FL-based approaches to solve problems in digital healthcare [18,19], security [20–24], e-commerce [25–27], etc.

Though the FL is a relatively novel research field, there is a need for systematization of the developed approaches devoted to different FL aspects. In this paper, the authors research existing approaches for intrusion detection based on federated learning in the context of the outlined challenges with a particular focus on architectural solutions and datasets used to evaluate the suggested approach. In [28], it is noted that FL-enabled IDS approaches for the Internet of Things (IoT) have just begun to develop, and the main *motivation* for this research is to understand what solutions for the problems mentioned above have been proposed, how efficiently they address them, and what obstacles are still needed to overcome. This could help researchers in the field of intrusion detection to define their goals more exactly, and determine experimental settings as well as techniques to face these challenges more clearly.

Thus, the contribution of the paper is as follows:

- The review of the architectures of the federated learning systems, including supported data partitions and requirements to the clients' availability and their computational resources.
- The review of the datasets that were used to evaluate the system and approaches to model federated settings.
- Comparative analysis of the proposed systems.

While there are multiple surveys on federated machine learning [6,15,18,29] that focus on different issues of the federated learning, there is a lack of surveys related to the intrusion detection systems based on federated learning. For example, in [15], the authors presented the most comprehensive survey on the open theoretical problems and challenges in FL. Rieke et al. [18] investigate possible usage scenarios of FL in healthcare systems, define typical FL settings, and discuss the research questions specific for this application domain. In [30], the authors focus on security and privacy issues in federated learning, and Novikova et al., in [29], evaluate the privacy-preserving mechanisms in FL systems and their applicability to driver and/or vehicle activity recognition. The closest survey to this one is [31]. The authors also review FL-based approaches to intrusion detection in IoT. However, unlike them, we focus on IDS architectural solutions and evaluate them in the context of the FL characteristics such as communication scheme and data partition, we evaluate the used datasets and achieved accuracy, and discuss what performance metrics are used. Thus, the *novelty* of the paper consists of the analysis of existing research, advantages, and open challenges of federated learning for intrusion detection.

The rest of the paper is organized as follows. A brief description of the federated learning systems is presented in Section 2. The typical architectural solutions for intrusion detection systems are given in Section 3. Section 4 discusses the research methodology and comparative criteria for the FL-based IDS systems and presents a detailed description of the selected approaches and their comparative analysis. The paper ends with a summary of the most important results of the research and guidance for future research.

2. Federated Learning Systems

The key idea of the FL consists of training local models directly on clients that generate or own data, then their parameters are aggregated to produce a global model that is distributed across all collaborating clients [1]. Thus, it is possible to outline three main components of the FL:

1. Clients that own data and train local model;
2. Server that coordinates the whole training process and computes the global model;
3. Communication environment.

FL systems are characterized by three important properties:

1. Communication scheme or FL topology [18];
2. Computational and network resources available to collaborating clients;
3. Type of the data partition.

The communication scheme defines how the federated learning process is orchestrated. The role of the coordinating server could be implemented by a designated server that is often represented by a trusted entity. This case corresponds to the centralized architecture of the FL systems. In the case of a decentralized FL system, the functions of the aggregating server are implemented by collaborating clients [6]. Figure 1 shows the communication schemes of the federated learning process. It is necessary to mention that there are also hybrid communication schemes that are represented either by a hierarchy of the decentralized federations or by a peer-to-peer network of federations with a centralized communication scheme [18].

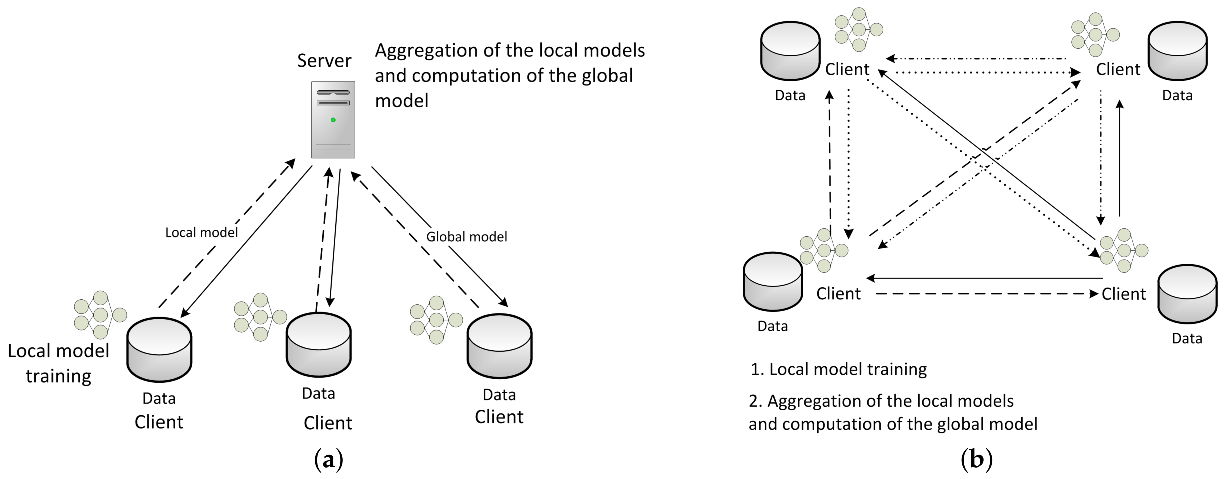
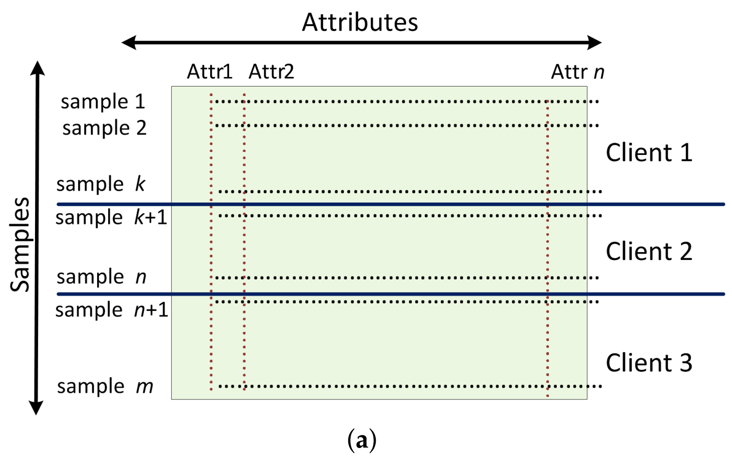


Figure 1. Overview of the communication scheme in FL systems: centralized communication topology (**a**), decentralized communication topology (**b**).

Depending on the client's computational and memory resources, and their network bandwidth characteristics, it is possible to outline cross-device and cross-silo settings. In cross-device settings, the clients have limited computational resources and low network bandwidth, their behavior is not stable, and they could drop out from the learning process. In contrast to the cross-device settings, the clients in cross-silo settings are characterized by a stable behavior and powerful computational and networking resources.

Type of the data partition defines how data are distributed across the clients. This notion is quite semantically similar to the notion of data partition commonly used in the distributed warehouses [32]. Usually, a dataset is characterized by two dimensions: (1) number of attributes and (2) number of samples. If the clients have similar sets of the attributes then the data are horizontally partitioned. In the case of vertically partitioned data, the clients have different data attributes for the same set of samples. This type of data partition is natural to many situations, for example, communication organization may have data about person's contacts, while the financial organization may have information about their financial state, and mining such data may give interesting insights into fraud detection activity. Figure 2 shows the schema of these data partition types. In the real world cases, the data could be partitioned across the clients partly vertically, partly horizontally—this case corresponds to the most complex type of the data partition, a hybrid one.



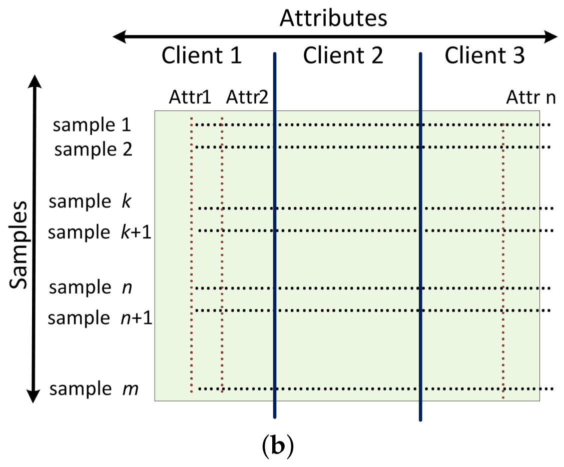


Figure 2. Types of the data partition: horizontally partitioned data (**a**), vertically partitioned data (**b**).

These FL properties define different types of the FL systems as well as the requirements to the throughput of the training process, choice of the aggregation function, etc.

FL paradigm is considered as a privacy-preserving computational paradigm because data are processed locally; they are not transferred to some entity in order to perform their processing. FL clients exchange model parameters only that are needed to form a global aggregated model. Thus, this reduces the risks of unauthorized access to the data when they are transmitted via communication channels or processed by the party. However, it was shown that there are theoretical FL-specific attacks that enable attackers to make inference on dataset characteristics and even recreate samples from it [29,30]. Such attacks are known as inference attacks and require capturing model parameters such as neural network gradients in order to perform it. Thus, the practical complexity of the inference attack is much higher than the attacks targeting data directly as it requires not only gaining access to model parameters but also applying specific techniques to analyze them. The latter depend on attacker's knowledge about trained analysis model, type of the analysis system, and their computational capabilities. Though inference attacks have high complexity, the additional privacy-preserving techniques have been proposed:

- Mechanisms based on differential privacy (DP);
- Data and model encryption techniques that include multi-party secure computations (MPC), homomorphic and functional encryption;
- Trusted execution environment (TEE).

The mechanisms based on differential privacy use random noise to mask user inputs, and, therefore, their application requires finding a balance between model accuracy and data owner privacy. They are suitable for the cross-device settings as they can tolerate client's dropouts.

Encryption-based techniques use different protocols to secure either input data or the whole training process. Their application significantly limits the choice of architectural solutions and places high demands on both computing resources and bandwidth of the computer network [29]. For example, the current solutions based on multi-party secure computations require two or three trusted computational entities to be set up in order to perform secure model training. Thus, this type of privacy-preserving mechanism is more suitable for cross-silo FL setting, the exception is constituted by the secure aggregation protocol proposed by K. Bonawitz et al. [33]. It is a lightweight encryption protocol that could be applied in cross-device settings.

TEE is a hardware-based solution; it provides a trusted environment for secure and verified code execution and strict data access control. TEEs are widely used in industrial IoT devices; thus, it could be a natural choice to design secured and privacy-protected industrial FL-based systems.

In [6], the set of design patterns, including privacy-enhancing techniques for FL architectures, is proposed. It includes three client management patterns (client registry, client selection, and client clustering), four model management patterns, three model training patterns, and four model aggregation patterns. The authors associate the patterns with the stages of the federated learning lifecycle and map them to the existing software or research papers.

3. Intrusion Detection Systems

An intrusion detection system (IDS) is an essential part of any information security system, and is aimed to detect attacks and anomalies in the information systems. Depending on the type of the input data, it is possible to outline host-based and network-based IDS. The network-based IDS monitors and analyzes the network traffic, while the host-based IDS collects and analyzes logs of the operating system and applications.

The typical architectural solution of the IDS includes components for data collection and processing, analysis, and attack (or anomaly) detection and response (see Figure 3). The basic components of the IDS also include a data repository that stores collected raw data and triggered alerts, and a knowledge base that contains information about attack detection rules, signatures, or malicious activity patterns. There are two main approaches to detect attacks or anomalies in the information system: signature-based and based on artificial intelligence (AI) [34]. The signature-based approaches detect attacks using pattern-matching techniques to find a known attack; that is why the attack knowledge base has to be kept constantly updated in order to preserve high level of the attack detection. This type of IDS demonstrates high performance in detecting known types of the attacks; to detect unknown or zero-day attacks, the AI-based approaches are adopted. Currently, researchers have proposed a wide variety of AI-based solutions including adaptive resonance theory [35], genetic algorithms [36,37], clustering [38], fuzzy logic [36], and deep networks such as convolutional neural networks [39,40], recurrent neural networks [41], deep auto-encoders [42], etc.

Further classification of the IDS is based on how these components are connected and coordinated in the system. Thus, there are monolithic, distributed, hierarchical, and agent-based systems [43,44]. The distributed IDS assumes that each node of the information system has its own IDS which are able to communicate with each other, and there is one dedicated IDS server that is responsible for the final data analysis and decision-making. In the concept of the hierarchical IDS, the local IDSs are grouped in clusters and each cluster has a selected head, a node that communicates with other clusters [45].

The architecture of the IDS is usually selected based on the type of the monitored information system and available computational, energy, and network bandwidth resources. For example, the typical IDS architecture for cloud intrusion detection is a distributed one. The primary reason for this choice is the necessity to analyze large volumes of network traffic, and acceleration of the computations on the data streams.

For the IoT-based systems, such as Smart Home systems, the recommended IDS architecture is a distributed hierarchical agent-based one [43]. This is explained by a fact that modern Smart Home systems are usually shipped with a set of cloud-based services provided by a product manufacture. In this case, the software IDS agent is installed on a home “smart” router using specialized middleware. Such an agent is capable of implementing different functions, including monitoring data flows from sensors, their preliminary analysis, and forwarding the analysis results to the master component located in the service provider cloud. The typical architecture of the Smart Home IDS is given in Figure 4. This architecture, on one hand, allows selecting a node with the more powerful resources; on the other hand, it supports fast preliminary analysis of the locally generated data with further advanced analysis implemented by a service provider. Another certain advantage of such approach is the ability to apply knowledge on the security incidents across different controlled Smart Home systems. The only serious drawback is significant privacy risks for the Smart Home users due to possible data leakage.

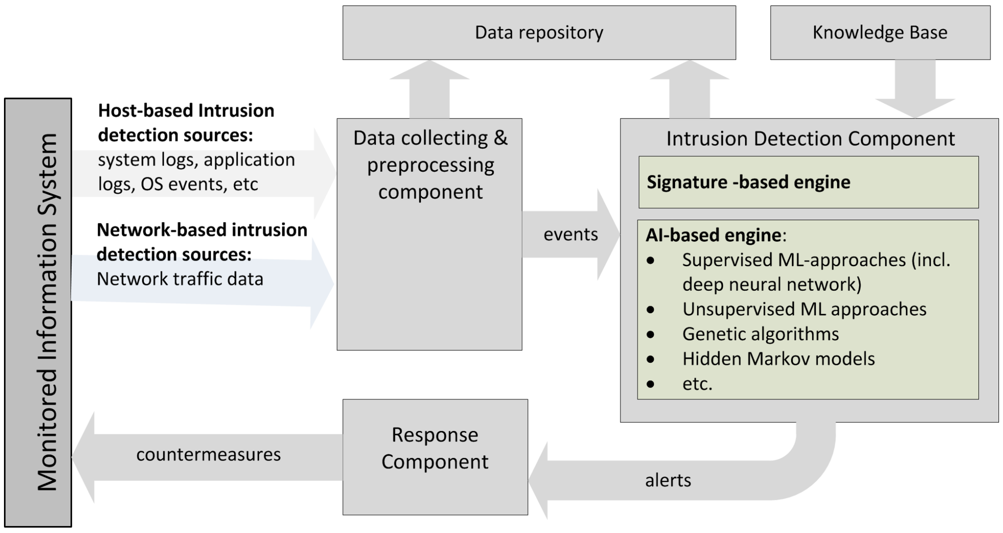

Data repository
Knowledge Base
Monitored Information System
Host-based Intrusion detection sources: system logs, application logs, OS events, etc
Network-based intrusion detection sources: Network traffic data
Data collecting & preprocessing component
events
Intrusion Detection Component
Signature -based engine
AI-based engine:
• Supervised ML-approaches (incl. deep neural network)
• Unsupervised ML approaches
• Genetic algorithms
• Hidden Markov models
• etc.
countermeasures
Response Component
alerts

Figure 3. Basic IDS architecture.

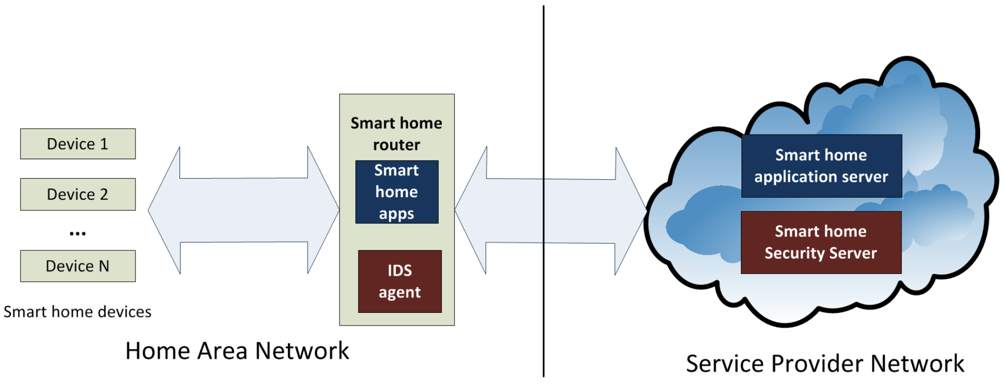


Figure 4. The high-level overview of the hierarchical distributed IDS for Smart Home systems.

The hierarchical distributed architecture for the IDS is also proposed for attack and anomaly detection in cyber-physical systems such as smart grids [46–48]. Typically, the smart grids are composed of different collaborating entities, including power stations, end users represented by Smart Home energy management systems, and legal controlling entities. The hierarchy of the local IDS systems could be constructed either based on type of the collaborating entity [46,47] or based on "intrusion boundaries" that limit the damage propagation in case of the successful attack [48].

Thus, it is possible to outline that modern IDS systems and FL-based systems share two common properties that define the choice of architectural solution: they are, namely, communication scheme (or topology) and requirements to the computational resources of the nodes with IDS agents. The architecture of the decentralized FL system correlates with the distributed IDS with peer-to-peer IDS agents, while centralized communication scheme of the FL system is close to the hierarchical IDS, in which local IDS agents or clusters of peer-to-peer agents communicate with the master IDS component to produce final decision. The characteristics of the cloud environment of the IDS are close to cross-silo FL settings, when the collaborating entities have enough computational and storage resources. An IDS for an IoT-based environment must consider similar constraints imposed by controlled devices as the FL system for cross device settings, i.e., power consumption, computing resources, and bandwidth availability.

There is no obvious matching for data partition property of the FL systems, though this property is extremely important when designing analytical system based on the FL principles. The majority of the existing FL frameworks support horizontally partitioned data [49]. In terms of the intrusion and anomaly detection, the network-based IDS could be considered as clients with horizontally partitioned data, as they typically operate with a certain set of attributes extracted from the network traffic, and the case of host-based IDS could be considered as the case of vertically partitioned data, as they operate with logs from different applications that have different format. Similarly, IDS agents in the cloud environment tent to have similar sets of analyzed attributes, while IDS agents deployed in the IoT-based settings need to process various features characterizing same objects as they collect data from a large variety of heterogeneous sensors.

The main benefit of the application of the FL technology for the IDS is the ability to reduce risks associated with sensitive and personal data processing. When the collaborating entities are presented by the commercial organizations, critical infrastructures, the usage of the FL increases the trust level between them and their security service provider as they also do not need to share their sensitive data, and at the same time they enable sharing knowledge about attacks and anomalies in the privacy-preserving manner. For the IoT-based environment, FL also allows reducing the network traffic which is an important factor in the condition of the energy-efficient networking and enables support of large scale of heterogeneous devices and sensors.

At the same time, the application of federated learning imposes certain requirements on the computational power of the entity that performs local model training, as well as on its storage capacity in order to store the data for training. Therefore, when designing a system, it is extremely important to evaluate the throughput of the FL-based analysis system. The need to consider these issues determined the main goals of this study, and the evaluation criteria and the results of the research are presented in the next sections.

4. Intrusion Detection Systems Based on Federated Learning

There are some reviews on federated learning for intrusion detection in Internet of Things. The theoretical and practical reviews can be outlined. For example, in [31], the challenges and future directions of FL-enabled IDS are considered, while in [28], the authors evaluate the FL-enabled IDS approach on an experimental basis.

The review provided in [31] is rather broad. The authors analyzed 15 research papers related to the FL-based IDS, but they are focused on the peculiarities of the proposed approaches and do not compare them nor outline the FL-specific parameters for comparison (i.e., FL architecture, ML model, dataset partition, aggregation function, etc.).

In [28], the authors outlined and compared 12 papers related to the application of FL to improve IDS efficiency. They considered attacks studied, datasets, ML model, FL implementation, aggregation function, training parties, and training rounds. However, the authors described them rather briefly and focused on the experimental evaluation of the FL-enabled IDS approach. Besides, while in [28] 12 papers are considered, only 5 of them use IoT datasets for the experiments [20–24].

In this paper, we selected for comparison the last papers related to the FL-based IDS. We searched for the “federated learning intrusion detection” keywords on the ScienceDirect website. As a result we found eight papers. We excluded two papers related to the poisoning attacks against FL-based systems [50,51] as in this paper we are interested in the existing FL-based intrusion detection solution. Finally we obtained six papers [28,52–56]. In addition, we added to the review several papers from the considered reviews that use IoT datasets for the experiments and are valuable for the FL-based intrusion detection area: [20–23,57]. The final list of the analyzed papers is [20–24,28,52–57].

We formulated the following research questions (RQ) for our review:

RQ1. Which FL architectures are used for the intrusion detection systems?
RQ2. Which data partitions and datasets are used to test the proposed solutions?
RQ3. Which types of attacks can be detected using the developed solutions?
RQ4. Which ML methods are used to detect attacks and/or anomalies?
RQ5. How do the authors implement their solutions?
RQ6. How do the authors test their solutions?
RQ7. What metrics do the authors use to validate their solutions?

Considering the formulated questions we outlined the following aspects while reviewing the papers:

- FL architecture, i.e., what communication topology (centralized or decentralized) is used;
- Data partition and dataset used;
- Attacks detected;
- ML method used to detect attacks and/or anomalies;
- Software implementation, including FL frameworks used;
- Conducted experiments;
- Used metrics and advantages.

In the paper, we are focused on the FL-based intrusion detection systems for IoT. We do not experimentally test the approaches to evaluate them. We review them considering the outlined aspects to highlight the open challenges and research directions. As there are no designated datasets for evaluating FL, the authors paid particular attention to how the data partition across the clients was modeled, and if the authors modeled the bias in the test data to evaluate its impact on the ML model efficiency.

4.1. Comparative Analysis of Federated-Learning-Based Intrusion Detection Systems

In this subsection we review the outlined papers considering the criteria specified above.

4.1.1. The Federated Learning-Based Intrusion Detection Systems for Smart Home Settings

There are several papers that consider Smart Home test cases.

An FL-enabled IDS approach based on a multiclass classifier [28].

In [28], the authors evaluate an FL-enabled IDS approach based on a multiclass classifier. They consider different data distributions for the detection of different attacks in an IoT scenario.

Federated learning architecture. The authors used centralized architecture. Namely, they applied an IBM framework for Federated Learning (IBMFL) [5]. It was set up for 10 clients that corresponded to 10 IoT devices and one aggregating server.

ML method. The authors applied a multi-class probabilistic classification model using multinomial logistic regression (softmax regression) [58].

Implementation. The authors used the IBM framework for Federated Learning IBMFL [5] and FedAvg [59] and Fed+ [60] aggregation methods. To implement the logistic regression algorithm, the Scikit-learn SGDClassifier (stochastic gradient descent) was used.

Dataset. The authors used the ToN_IoT dataset for the experiments [61]. They implemented three different partitions for the selected ToN_IoT dataset: (1) based on the destination IP address; (2) balanced the data considering the attacks across the clients; (3) hybrid approach, where the authors used the Shannon entropy to find the compromise between the attack types balance and the destination IP address [62]. The considered attacks are backdoor, DoS, DDoS, injection, MITM, password, ransomware, scanning, and XSS.

Experiments. The authors used a Lenovo laptop with an AMD Ryzen 7 4800H with Radeon Graphics, and 16 GB of RAM for the experiments. Based on the different data distributions the authors created three scenarios of the experiments, namely, basic scenario using the partition (1), the balanced scenario using the partition (2), and the mixed scenario using the partition (3). The model was trained on 300 rounds for each scenario with aggregation after each epoch. Additionally, the authors measured the accuracy of the model trained by each client independently from other parties in a distributed scenario. Finally, they measured the accuracy of the model trained in centralized mode that is equal to 0.724 using multinomial logistic regression.

The results of the experiments demonstrated that Fed+ provides higher average accuracy than FedAvg for the considered federated scenarios. For the basic scenario, accuracy is about 0.8725 using Fed+ and 0.8718 for the distributed method. For the balanced scenario, the accuracy is 0.9039 for Fed+ and 0.9065 for the distributed setting. For the mixed scenario, accuracy using Fed+ is 0.8869 and 0.877 for the distributed case.

Used metrics:

- Accuracy, which is calculated as ratio of (true positives + true negatives) to the (true positives + false positives + false negatives + true negatives);
- Precision, which is calculated as ratio of true positives to the (true positives + false positives);
- Recall, which is calculated as ratio of (true positives) to the (true positives + false negatives);
- F1-score, which is calculated as ratio of (recall $\times$ precision) to (recall+precision) multiplied on 2;
- False positive rate (FPR), which is calculated as ratio of false positives to the (false positives + true negatives).

Advantages:

- The impact of non-independent and identically distributed data is considered;
- Different aggregation methods are considered;
- Different data distributions are considered;
- Different training rounds are considered;
- The attacks detection is considered to deploy the most effective countermeasures dynamically;
- The Fed+ is used firstly for the FL-enabled IDS for IoT.

Limitations:

- The research is only focused on the impact of different data distributions;
- The network traffic from IoT_ToN dataset is used without consideration of the IoT devices telemetry.

A federated self-learning anomaly detection system for IoT—DÏoT [20].

In [20], a federated self-learning anomaly detection system for IoT—DÏoT—is presented. The authors use the federated learning approach for the intrusion detection based on anomalies in the device type communication behavior. To detect anomalies, the authors represent the network packets as symbols and use language analysis techniques.

Federated learning architecture. The authors implemented the centralized architecture. The IoT devices are connected to the local security gateways that implement an anomaly detection service that trains models locally; these models are aggregated in the global detection model by the external IoT security service connected to multiple gateways.

The federated learning process is organized as follows. In the first step, each security gateway requests the initial gated recurrent unit (GRU) model for the specific IoT device profiles (*type#k*) from the IoT security service (initially the model is random) and obtains it in the second step. In the third step, the security gateways retrain the global model locally. Then, in the fourth step, the security gateways send the model updates to the IoT security service. It aggregates the local models to enhance the global model [1]. In the final step, IoT security service shares the updated global model for *type#k* devices with security gateway and uses it for anomaly detection.

ML model. The authors used the gated recurrent unit (GRU) model.

Implementation. To implement the described architecture, the following tools are used:

- Server-side application—the flask [63] and flask_socketio [64] libraries;
- Client-side application—the socketIO-client [65] library and the gevent asynchronous framework [66], which provides a clean API for concurrency and network related tasks;
- Gated recurrent unit (GRU) network—the Keras [67] library with Tensorflow backend;

Dataset. The authors collected three datasets—activity dataset, deployment dataset, and attack dataset—using a laboratory network. It incorporated a laptop with Kali Linux where hostapd was used to create a gateway acting as an access point with WiFi and Ethernet interfaces. IoT devices were connected to this access point. The authors used tcpdump on the gateway to collect the network traffic packets from the IoT devices.

The activity dataset incorporates traffic representing device activity for 33 IoT devices (i.e., IP cameras, smart power plugs and light bulbs, sensors, etc.) as well as traffic collected during two to three, during which actions were triggered only occasionally.

The deployment dataset was collected from 14 Smart Home IoT devices installed in a realistic Smart Home deployment setting during one week.

Attack dataset was collected for the devices infected with Mirai malware [3] including the traffic when Mirai was in a standby mode.

Used aggregation method. Global model aggregation:

$$G = \sum_{i=1}^{n} \frac{s_i}{s} W_i, \tag{1}$$

where n—number of clients; $W_1, \ldots, W_n$—clients' associated model weights; $s_1, \ldots, s_n$—associated number of data samples used for training; $s = \sum_{i=1}^{n} s_i$.

Used metrics:

- False positive and true positive rate;
- Average detection time: 257 ± 194 ms;
- Processing performance of GRU—average processing time per symbol (packet) for prediction—0.081 ($\pm$0.001) ms for the desktop utilizing its GPU and 0.592 ($\pm$0.001) ms when executed on the laptop with CPU;
- On average, training a GRU model for one device type took 26 min on the desktop and 71 min on the laptop hardware.

Advantages:

- Implemented;
- Collected dataset of network traffic of test IoT devices communication behavior (33 devices, 23 types), dataset of consumer IoT devices (14 devices), dataset of infected with Mirai malware (5 devices) [3];
- Experiments with real IoT devices;
- For Mirai malware: 95.6% detection rate and $\approx$257 ms at detecting compromised devices;
- No false alarms;
- Does not require any human intervention or labeled data to operate;
- Learns anomaly detection models autonomously, using unlabeled crowdsourced data captured in client IoT networks.

A framework based on federated learning for detection of malware in IoT devices [22].

In [22], the authors propose the framework based on federated learning for detection of malware in IoT devices.

Federated learning architecture. The authors implement the centralized architecture. The proposed framework incorporates clients (with single device per client) and a server. The server implements the following functionality: initializes the model; aggregates the models sent by the clients into a global model; coordinates the collaborative normalization, the collaborative grid searches, and the collaborative threshold selection. It is cross-silo FL when there are few powerful and reliable federated clients.

ML methods. The authors used supervised and unsupervised federated models (multilayer perceptron and autoencoder). They tested different number of hidden layers and hidden neurons. After each hidden layer, the exponential linear unit (ELU) activation function was used. The model updates were computed using stochastic gradient descent (SGD).

Implementation. For aggregation, the authors used mini-batch aggregation and multi-epoch aggregation from the FedAVG. The training of multi-epoch aggregation was repeated for 30 rounds.

Dataset. The authors used the N-BaIoT dataset that models network traffic of nine real commercial IoT devices of different types [68]. The malware threat is considered (the devices are uncorrupted or infected by Mirai or BASHLITE). In the supervised case for each device, the data were split chronologically between three parts: the train set—79%, the unused set (the authors left a small set of samples unused between the train part and the test part for each file in the dataset as soon as the features of packets captured in a very short time interval are highly correlated)—1%, and the test set—20%. In the unsupervised case, the benign data were split between four sets: the train set—39.5%, a threshold-selection set—39.5%, the unused set—1%, and the benign part of the test set—20%.

The authors also outline three data scenarios via the dataset rebalancing to balance the number of samples and the proportions of classes for the devices. In the first case, they saved the original dataset balance for every device: 7.87% benign traffic and 92.13% attack traffic. In the second case, they outlined for every device 50% benign traffic and 50% attack traffic. In the third case, the authors outlined 95% benign traffic and 5% attack traffic.

In addition, the authors selected for each device the same number of samples—100,000 samples for the supervised solution and 10,000 for the unsupervised one.

Experiments. The authors evaluated the performance of FL models (federated with mini-batch avg and federated with multi-epoch avg). In addition, they compared it to two traditional approaches (naive decentralized approach and centralized approach). For the experiments, the authors used eight clients, each owning data of one of the nine devices from the N-BaIoT dataset, while one device was kept for testing.

For the supervised solution, the training was conducted for four epochs, for the unsupervised solution—120 epochs. The experiments were repeated five times; thus, the results of each experiment represent the average over 45 runs (nine devices and five executions).

For the supervised methods, the experiments showed that the centralized method's performance was higher than the distributed one. The results obtained for mini-batch avg were very close to the centralized ones. Multi-epoch avg showed an insignificant decrease in the accuracy on known devices and an accuracy exceeding the centralized one on the new device.

For the unsupervised methods, the experiments showed that centralizing the data gives higher performance than the naive method, while the multi-epoch avg and the mini-batch avg methods demonstrate the results close to the centralized one.

The authors also considered the data poisoning attacks and their impact on the model performance and demonstrated that it is reduced substantially.

Used metrics:

- True positive rate, which is calculated as ratio of true positives to the (true positives + false negatives);
- True negative rate, which is calculated as ratio of true negatives to the (true negatives + false positives);
- Accuracy, which is calculated as ratio of (true positives + true negatives) to the (true positives + false positives + false negatives + true negatives);
- F1-score, which is calculated as ratio of true positives to the (true positives + 1/2 of (false positives + false negatives)).

Advantages:

- Implemented (code is available at [69]);
- The N-BaIoT dataset was used for the experiments;
- Preserves the security and privacy of the model;
- Obtained results for FL methods are close to the results obtained using centralized models while preserving the privacy;
- The malware threats are considered;
- The cyberthreats against federated learning framework are considered;

- Different data partitions are investigated.

Limitations: performance is reduced substantially in case of data poisoning attacks; scalability is not considered due to the dataset limitations.

FL-based edge cloud architecture for attack detection [23].

In [23], the authors propose an edge cloud architecture for attack detection.

Federated learning architecture. The authors implement hierarchical architecture in the cloud. The proposed architecture incorporates the following layers: data perception layer (IoT devices with sensors); edge layer (IoT gateways responsible for normalizing the data and containing LocKedge module); network layer (secures data transfer); data management layer (the cloud that is responsible for deciding the number of neurons per layer and the weights of the neural network algorithm); application layer (applications management); business layer (IoT applications and services management). Thus, the detection module is implemented at the edge in the IoT system.

ML methods. LocKedge (Low-Complexity Cyberattack Detection in IoT Edge Computing) based on traditional neural network (NN) that performs multi-class classification to detect different types of attacks. The authors used the principal components analysis (PCA) method to extract the most important features. For the neural network model, the authors chose the ReLU as activation function in hidden layers and softmax activation function for the multi-class classification problem.

Implementation. The authors implemented their solution both in centralized and federated learning mode. In the federated learning mode, the edge nodes implement detection and training. They send weights to the cloud to update the global training model. Thus, the workload of the central cloud is reduced.

The authors used a Raspberry Pi 3B+ to implement the edge gateway (1.4 GHz ARM-based quad-core processor with 400% CPU usage at maximum and 1 GB RAM) with installed Raspberry Pi OS. They used the Python 3 programming language to implement the detection solution.

Dataset. The authors used the BoT-IoT dataset [70] (5% extracted from the original dataset). The dataset is generated considering the following IoT devices: a weather station, a smart fridge, remotely activated motion activated lights, and a smart thermostat. It includes the following types of attacks: DDoS (HTTP, TCP, UDP), DoS (HTTP, TCP, UDP), OS fingerprinting, server scanning, keylogging, and data exfiltration attacks.

For the edge case, the authors divided the BoT-IoT dataset into four smaller client datasets according to the source IP address. Each dataset then was divided into a training and testing set.

Experiments. The authors compared the complexity of traditional neural network and the proposed LocKedge method. The experiments for different number of the neurons showed that that the neural network complexity is higher than LocKedge complexity, while the training time of LocKedge is lower than neural network.

Then, the authors compared the performance of the centralized LocKedge with the neural network model (without the feature processing). They varied the number of neurons in the hidden layer from 6 to 46. The accuracy of the centralized LocKedge (about 0.999) is higher than NN (0.997).

The authors also evaluated the average detection rate for each attack type using LocKedge, NN, DNN, recurrent neural network (RNN), and convolutional neural network (CNN). The comparison shown that the LocKedge demonstrates results better than, or close to, other methods, except the DoS-HTTP and data theft attack. The authors also compared the centralized LocKedge with such machine learning algorithms as K-nearest neighbors (KNN), decision tree (DT), random forest (RF), and support vector machine (SVM). The average detection rate of the centralized LocKedge is higher than the other methods in most classes.

According to the paper, the precision of the centralized LocKedge, as well as F1-score, are higher than the values obtained by the other solutions.

The authors also analyzed the performance of the training and detection at the edge. They performed feature extraction using joined training dataset, while the detection models were trained separately. They evaluated the resulting global model using four different test datasets after each communication round. The number of communication rounds is 1000 and the number of local epochs was set to 1. The accuracy stopped increasing and the loss stopped decreasing after about 350 communication rounds, giving results comparable to the centralized approach. In terms of F1-score, detection rate, and precision, some types of attacks (DoS-HTTP, DDoS-HTTP and theft-data) give worse results than in centralized mode.

Finally, the authors evaluated the edge computing capacity. The authors loaded traffic of 400 to 2400 samples per second to the PI3. The rate of 2400 samples per second leads to 100% of the CPU usage of a core among a quad-core. The authors also found out that the CPU usage increases exponentially as the attack rate increases, and they conclude that the maximum attack rate for the Pi3-based edge is 9600 samples per second. The memory usage of the PI3 also increases with the attack rates growth. It can process up to 1800 samples per second.

Used metrics:

- Complexity of the algorithms;
- The training time performance (the time of reading data for the training phase);
- Accuracy (the total number of correctly predicted samples in all tests);
- Detection rate (DR) (the number of the actual positives that are predicted as positive);
- Precision (the ratio of true positives to the (true positive + false positive);
- Recall (the ratio of true positives to the (true positive + false negative);
- F1-score (2 multiplied by the ratio of (precision * recall) to the (precision + recall);
- CPU and RAM usage.

Advantages:

- Novel attack detection mechanism LocKedge is introduced; the experiments show the higher performance of the proposed mechanism compared to other ML methods;
- The real traffic BoT-IoT dataset is used;
- The edge computing capacity is evaluated.

Limitations: detection rate of DoS-HTTP, DDoS-HTTP, and theft-data attacks is lower than for other types of attacks.

A distributed ML-based IDS between organizations [57,71].

In [57,71] authors proposed a hierarchical distributed IDS based on FL. Sarhan et al. [57] adopted FL to design a distributed ML-based IDS between organizations. The two-level hierarchical approach is suggested to cope with heterogeneity of the IT ecosystem of the organizations and to enable knowledge sharing between organizations even in the case that they do not trust each other. In order to achieve the letter requirement, the smart contract running on a permissioned blockchain is adopted. The training process includes two levels of aggregation: local aggregation, when the aggregation is performed on the intermediate level, and the global one that is implemented by a global server.

Federated learning architecture. The proposed architecture includes two levels of the hierarchy; the top level is represented by a global server that is running a blockchain smart contract. This entity is also known as the reducer. It is hosted on a cloud-decentralized blockchain, and runs FL tasks using a smart contract. The intermediate level is represented by a combiner or aggregating server of the collaborating organizations. The combiner aggregates local models from the smart IoT endpoints that perform local models directly on data.

ML methods. The trained model is a deep feed-forward (DFF) neural network with the following architecture: input layer that consists of neurons equal to number of data features, four middle layers with Relu activation function, and an output layer that consists of one sigmoidal neuron.

ML methods. Aggregation function is FedAvg.

Implementation. Custom.

Dataset. NFBoT-IoT-v2 with NetFlow 9 attributes constructed from BoT-IoT dataset [70,72] with four types of attack: DDoS, DoS, reconnaissance, and theft.

Experiments. The authors considered two experimental scenarios. One scenario corresponds to the collaborating entities. The authors modeled the cooperation of two organizations with two smart endpoints each. Thus, to model partition across four collaborating endpoints, the authors split the dataset in such a manner that each client had malicious samples of at least two attack types. Another scenario corresponds to a non-collaborating entity. In this scenario, the training of the intrusion detection model was performed on the part of the dataset that belongs to the selected party.

Used metrics:

- Accuracy;
- Detection rate;
- False alarm rate (FAR)—the percentage of benign data samples incorrectly classified;
- F1-measure.

Advantages:

- Hierarchical FL-based IDS supports large-scale deployments and solves the problem of aggregation of large number of local updates from the numerous IoT endpoints.
- Some types of the "unseen" attacks are reliably detected. However, this is true only for DDoS and DoS attacks.
- Application of the blockchain smart contracts provides a defense mechanism against data/model poisoning attacks.

Blockchained-federated-learning-based cloud intrusion detection scheme for IoT [53].

Federated learning architecture. The architecture of the proposed solution includes four layers—application layer, federated-learning layer, chaincode layer, and blockchain layer. The architecture is centralized.

Privacy-preserving mechanism. Hyperledger fabric (blockchain).

ML method. A semi-supervised learning algorithm based on divergence for local model training (decision tree and multilayer perceptron) and FedAvg algorithm for global model training.

Implementation. Implemented (custom).

Dataset. KDDCup99 dataset.

Experiments. The authors deployed the testbed for the experiments using Ubuntu operating system. They used hyperledger fabric for the blockchain project. Four hosts simulating different organizations are connected to a blockchain network; three of them simulate regional service parties (for local models), while the last simulates global service party (for global model). The authors compared their solution—semi-supervised learning algorithm based on divergence for local model training (decision tree and multilayer perceptron)—with random forest and SVM (50-iteration training). They concluded that the multilayer perceptron is the most suitable for real scenarios. After that, the authors compared the results of the global model with CNN and DNN (100-iteration training) using AUC. The average AUC of the three models are 0.908, 0.849, and 0.899, respectively.

Used metrics:

- TP, FN, FP, and TN;
- Accuracy;
- Precision;
- Recall;
- F1-score;
- AUC (area under curve)—the area of ROC curve;
- Time.

Advantages:

- Implemented;
- Preserves the security and privacy of the model;

- Describes and analyzes privacy-preserving mechanism;
- Discusses the optimal block file segmentation size in the fabric storage scheme (64 KB);
- Outperforms other models selected for comparison considering the selected metrics;
- The following threats are considered: DoS, Probe, R2L, and U2R.

Limitations: While the paper describes the IDS for IoT, the dataset selected for the experiments is not the IoT dataset. Data partition is not considered.

4.1.2. The Federated-Learning-Based Intrusion Detection Systems for Industrial Cyber-Physical Systems

Some researches consider industrial cyber-physical systems case.

A federated deep learning for intrusion detection in industrial cyber-physical systems [21].

In [21], the federated deep learning is used for intrusion detection in industrial cyber-physical systems. The authors propose a deep learning model based on a convolutional neural network and a gated recurrent unit designed for detection of the denial-of-service, reconnaissance, response injection, and command injection attacks. In addition, the authors develop a federated learning framework and a Paillier public-key cryptosystem-based secure communication protocol to ensure the security and privacy of model parameters. The proposed intrusion detection scheme incorporates the following stages: (1) system initialization; (2) local model training by industrial agents; (3) model parameters encryption by industrial agents; (4) model parameters aggregation by the cloud server; (5) local model updating by industrial agents.

Federated learning architecture. The proposed framework includes three types of entities, namely, a trust authority (it bootstraps the whole system, generates public and private keys, and establishes secure communication channels), a cloud server (builds a comprehensive intrusion detection model), and industrial agents (builds a local intrusion detection model based on its own data and updates the parameters of the intrusion detection model via interaction with the cloud server).

Implementation. The authors implemented the proposed attack detection model using the Keras API [73]. The federated learning framework was built using Python framework Flask.2. The authors used for the experiments the Ubuntu 18.04.3 LTS platform with an Intel Xeon E5-2618L v3 CPU and an NVIDIA GeForce RTX 2080TI GPU (64 GB RAM).

Dataset. The authors used a gas pipelining system's dataset for the experiments [74]. The authors divided the dataset into training part (80%) and testing part (20%). They further divided the training part into even partitions for the industrial agents to implement local model training. The authors considered the following attacks: DoS, DDoS, command injection, response injection attacks, reconnaissance attacks, response injection attacks, and command injection attack.

Experiments. The authors implemented Schneble's [75], Nguyen's [20], and Chen's [76] models and compared them with their solution. In addition, the authors compared them with the local intrusion detection models built by each industrial agent and with the intrusion detection model generated in the centralized mode. The authors used 3, 5, and 7 industrial agents for the experiments and 2, 4, 6, 8, and 10 communication rounds. They trained all models on the same testing data. The experiments demonstrated that the proposed intrusion detection model allows obtaining higher performance than other models selected for comparison, as well as the local intrusion detection models. Furthermore, its performance is close to the ideal model results.

Used metrics:

- Accuracy = 99.20% for three industrial agents, 99.20% for five industrial agents, and 99.20% for seven industrial agents;
- Precision = 98.86% for three industrial agents, 98.85% for five industrial agents, and 98.85% for seven industrial agents;
- Recall = 97.34% for three industrial agents, 97.45% for five industrial agents, and 97.47% for seven industrial agents;

- F-score = 98.08% for three industrial agents, 98.13% for five industrial agents, and 98.14% for five industrial agents.

Advantages:

- Implemented;
- Preserves the security and privacy of the model;
- The experiments are performed on a real industrial cyber-physical systems (CPS) dataset;
- Outperforms other models selected for comparison considering the selected metrics;
- The following threats are considered: DoS, DDoS, command injection, response injection attacks, reconnaissance attacks, response injection attacks, and command injection attack;
- The cyberthreats against a federated learning framework are considered: eavesdropping of data resources, eavesdropping of model parameters.

Limitations: Limited with industrial CPSs. Unfortunatly, the paper does not provide the experiment results by attack types.

A collaborative learning model for attack detection in industrial IoT [24].

In [24], the collaborative learning model for IoT is proposed. The cores of the approach are the smart "filters" for attack detection and prevention on IoT gateways.

Federated learning architecture. The authors propose to use data collected from the network to train the model on the gateway and then share it with other gateways using a center server that aggregates the models. Thus, the authors use centralized architecture.

ML model. Deep neural network to train models on the gateways and the average gradient update algorithm to aggregate these local models into the global model.

Implementation. Prototype for the experiments.

Dataset. KDD [77], NSLKDD [78], UNSW-NB15 [79], and N-BaIoT [68].

Detected attacks. The authors used an anomaly-based approach; thus, they demonstrate an ability to detect the attacks from the selected datasets with rather high accuracy, namely, denial-of-service (DoS), attack from remote to local machine (R2L), unauthorized access to local administrator user (U2R), probing attack, Mirai, and BASHLITE.

Experiments. The authors distribute the dataset into different subnetworks (two and three) to train the local models. Using the selected datasets, the authors conducted the experiments using their intrusion detection solution and the following methods for comparison: centralized learning model for classification and anomaly detection, K-nearest neighbors classifier, K-means, decision tree, multilayer perceptron, logistic regression, and support vector machine. According to the provided results of the experiments, the proposed solution outperforms the other methods on the selected datasets.

Used metrics:

- True positive (TP), true negative (TN), false positive (FP), and false negative (FN).
- Accuracy (ACC), which is calculated as follows:

$$ACC = \frac{1}{M+1} \sum_{m=1}^{M+1} \frac{TP_m + TN_m}{TP_m + TN_m + FP_m + TFN_m}, \tag{2}$$

  where $M+1$—the total number classes for normal and attack traffic. The obtained values are as follows: for two subnets: 97.52% for the KDD dataset, 93.99% for the NSL-KDD dataset, 95.6% for the UNSW dataset, and for N-BaIoT dataset from 67.53% to 99.84%, depending on the IoT device; for three subnets: 97.54% for the KDD dataset, 93.37% for the NSL-KDD dataset, 95.67% for the UNSW dataset, and for N-BaIoT dataset from 67.27% to 99.84%, depending on the IoT device.
- Privacy.
- Time.

Advantages:

- Implemented;
- Preserves the security and privacy of the model;

- Outperforms other models selected for comparison considering the selected metrics;
- Anomaly-based and thus can detect attacks of various types;
- The performance in terms of accuracy, privacy, and time is evaluated.

Limitations: Unfortunately, the paper does not provide the experiment results by attack types. While the approach is proposed for Industry 4.0, only one from the tested datasets is devoted to IoT and it contains traffic from the Smart Home IoT devices. While the authors conduct the performance evaluation, they do not provide the characteristics of the testbed for the experiments. Unfortunately, it is also impossible to evaluate privacy and time performance results as soon as the metrics used are not explained. The details of the privacy-preserving mechanism and dataset partition are not provided.

FL-based intrusion detection in cyber-physical settings in the example of the water treatment facility [4].

In [4], authors also study the problem of the intrusion detection in cyber-physical settings in the example of the water treatment facility. The key distinction from the previous research is that the authors focus on the case of vertically partitioned data and propose a way to model it by grouping readings from the sensors on the basis of the stages of the technological process.

Federated learning architecture. The authors propose to supplement every technological process with an intelligent agent-hub that not only registers data from sensors but also performs local training using them. These hubs are coordinated by an additional aggregating server located together with the central server of the SCADA system. This server is also responsible for controlling the global model performance and initiation of its training process.

Implementation. To implement the proposed approach, the authors used Federated AI Technology Framework [80] that is considered by the IEEE P3652.1 Federated Machine Learning Working Group as a standard-setting FL framework [15].

ML model. The choice of the ML model was defined by the algorithms available in FATE framework for vertically partitioned data. The authors evaluated gradient boosting decision trees (GBDT) with Paillier homomorphic encryption.

Dataset. The SWAT2015 dataset [81] was used in the experiments. This dataset models the functioning of the secure water treatment facility during 12 days and it contains both network data and data from sensors. In [4], a dataset with readings from physical sensors was used. To model vertically partitioned data, it was split into five groups based on technological process.

Detected attacks. The SWAT dataset contains a set of attacks that are targeted to inject control signals in order to manipulate the system state and cause the sensor's degradation.

Experiments. The authors performed two series of the experiments in which the goal was to compare the efficiency of the federated learning in CPS use case. In the first series, the authors modeled the FL settings with five clients with vertically partitioned data, and in the second case, they performed experiments on the initial dataset to evaluate the impact of the FL.

Used metrics:

- Accuracy;
- Time of the training and inference time.

Advantages:

- Preserves the security and privacy of the model as the additional homomorphic encryption is implemented to secure clients inputs;
- The accuracy of GBDT in FL mode is comparable with GBDT in centralized mode, reaching 99% of the accuracy;
- The performance in terms of accuracy, privacy, and time is evaluated.

Limitations:

- Major limitation of the approach is the duration of the inference—it takes approximately 40 min, which is unacceptable for intrusion detection;

- The choice of the metrics assessing the models are limited by the FATE framework;
- The aggregation algorithm does not tolerate clients' dropouts.

4.1.3. The Federated-Learning-Based Intrusion Detection Systems for Specific Areas

While most papers consider Smart Home infrastructures or industrial IoT architectures, there are also specific test cases, such as agricultural IoT or Internet of Medical Things.

A federated-learning-based intrusion detection system for agricultural IoT infrastructures [52].

In [52], the authors introduced a federated-learning-based intrusion detection system for agricultural IoT infrastructures.

Federated learning architecture. The authors used the centralized architecture where the devices train local models of the same structure using different local datasets. They share the updates from their model with an aggregation server that produces an improved detection model. The developed FELIDS system protects data privacy through local learning.

ML methods. The FELIDS uses deep neural networks, convolutional neural networks, and recurrent neural networks. The authors provide the following parameters for the global classifier: batch size = 100; local epochs = 1; global epochs = 50; learning rate = 0.01–0.5; regularization is L2; global loss function is categorical_crossentropy; activation function is ReLu; classification function is softmax, and optimizer is Adam.

Implementation. The authors used Google Colaboratory [82] for the experiments and the Python 3 programming language. To implement the proposed system the authors used NumPy, Pandas, TensorFlow, Keras, Scikit-learn, and SMOTE libraries. The Sherpa.ai FL framework [83] was used for the federated learning. To evaluate the training energy consumption, the authors used carbontracker [84].

Dataset. The CSE-CIC-IDS2018, MQTTset, and InSDN real-world traffic datasets are used for the experiments. The authors considered the following types of attacks: conventional network-based attacks, namely, denial-of-service (DoS), distributed denial-of-service (DDoS), brute force, web-based, infiltration, and botnet (CSE-CIC-IDS2018 daaset); IoT protocol-based attacks, including DoS, BruteForce, Malformed, SlowITe, and Flood (MQTTset that contains message queue telemetry transport (MQTT)-protocol-based communications between IoT devices); sophisticated network-based attacks, including network and application DoS attacks, DDoS attacks, password guessing, web applications-based attacks, probes, botnets, and U2R attacks (InSDN dataset).

The authors used the independent and identically distributed (IID) data partition among the clients. In this case, the data distribution for the clients is consistent with the data distribution in the entire dataset. They also used non-independent and identically distributed (non-IID) data partition among the clients. In this case, the data distribution for the clients is not consistent with the data distribution in the entire dataset.

Experiments. The authors first conducted the experiments for the centralized model and obtained the best results for the InSDN dataset: 98.54% accuracy for DNN, 97.71% for CNN, and 97.84% for RNN. They obtained the worst results for the MQTTset dataset: 90.40% accuracy for DNN, 90.76% for CNN, and 90.05% for RNN.

The authors provide the following learning scenario based on the FedAvg algorithm: (1) a generic neural network model is generated by the FELIDS server (the set of initial model weights, the neural network architecture, the hyperparameters, and local and global epochs are identified); (2) the clients download the generic model from the FELIDS server; (3) the generic model is retrained by the clients locally using their private data computing new local set of weights; (4) the clients share the updated model parameters with the server; (5) the FELIDS server aggregates the parameters and creates an updated global model; (6) the updated global model parameters are shared by the server with the clients; (7) the clients use the updated parameters to retrain the model using new local data.

The authors used different sets of clients: 5, 10, and 15. A total of 50 communication rounds were used.

It should be noted that a secure gRPC channel is used for communication between the clients and the server.

The results of the experiments show that the FELIDS system accuracy is close to the centralized model after 50 rounds, but it preserves the privacy of the client's data. It also should be noticed that there is a gap in accuracy between the best and worst clients, especially for the non-IID case.

Additionally, in the paper, the time complexity (linear, depends on the number of clients, layers, number of local epochs, number of local examples for the client and aggregated local models parameters) and power consumption (quadratic complexity, depends on the total federated rounds, the total number of clients, the time, the energy power of client and server) were evaluated. It was tested on the experiments using the Intel CPU Core i5-6300U @ 2.4 GHz, 8.00 GB of RAM, and Ubuntu 20.04.3 LTS. The experiments showed that the time and energy consumption grow with number of clients and rounds, but do not depend on the data distribution technique. According to the experiments, the most efficient method in terms of time and energy consumption is DNN.

Used metrics:

- Time complexity;
- Power consumption;
- True positive (the number of correctly classified as attacks attack samples);
- False positive (the number of wrongly classified as attacks benign samples);
- True negative (the number of correctly classified benign samples);
- False negative (the number of wrongly classified as benign attacks samples).
- Accuracy (the ratio of correct classifications number to the total input number), which is calculated as the ratio of (true positive + true negative) to the (true positive + true negative + false positive + false negative);
- Precision (the ratio of correct attack classifications to the total number of attack results predicted), which is calculated as the ratio of true positive to the (true positive + false positive);
- Recall (the ratio of correct attack classifications to the total number of all samples that should have been identified as attacks), which is calculated as the ratio of true positive to the (true positive + false negative);
- F1-score (the harmonic mean between precision and recall), which is calculated as the ratio of (precision $\times$ recall) to the (precision + recall) multiplied by two.

Advantages:

- Implemented;
- The MQTTset dataset that contains MQTT-protocol-based communications between IoT devices is used for the experiments;
- The time complexity and power consumption are calculated;
- The time complexity and power consumption are evaluated on the experiments;
- Two data partition cases are considered;
- The results are close to the centralized model's accuracy while preserving data privacy.

Limitations: Limited application area—agricultural Internet of Things; the gap between the performance results for the best and worst clients, especially for the non-IID case.

The results of comparison are summarized in Table 1.

Table 1. Comparison of the federated learning architectures.

Ref.	FL Architecture	Dataset	Attacks	Data Partition	ML Method	Implemen-Tation	Conducted Experiments and Best Accuracy	Used Metrics
[20]	centralized	generated Smart Home dataset	Mirai malware	by device type	language analysis techniques, GRU	flask, flask_ socketio, socketIO-client libraries, gevent asynchronous framework, Keras library with Tensorflow backend	FL-scenario anomaly detection performance and time, 95.6% detection rate	FP, TP, average detection time, processing performance of GRU
[28]	centralized (10 clients)	ToN_IoT	Backdoor, DoS, DDoS, Injection, MITM, Password, Ransomware, Scanning, XSS	by IP address, balanced considering the attacks, hybrid approach	multinomial logistic regression (scikit-learn SGDClassifier)	IBMFL, FedAvg and Fed+	centralized mode (0.724), three FL-based scenarios (basic—0.8725, balanced—0.9039, hybrid—0.8869) and distributed method (basic—0.8718, balanced—0.9065, hybrid—0.9065)	accuracy, precision, recall, F1-score, FPR
[22]	centralized (8 clients), cross-silo FL	N-BaIoT	Mirai or BASHLITE	chronologically; original, balanced by attacks, 5% attack traffic	multilayer perceptron and autoencoder	FedAVG, 30 rounds of training	4 epochs of training for supervised solution and 120 epochs for unsupervised, close results to the centralized method	TP, TN, accuracy, F1-score
[23]	hierarchical architecture in the cloud	BoT-IoT	DDoS (HTTP, TCP, UDP), DoS (HTTP, TCP, UDP), OS Fingerprinting, Server Scanning, Keylogging, Data exfiltration	by source IP address	LocKedge (NN based), PCA to extract features	Raspberry Pi 3B+ for the Edge gateway, Raspberry Pi OS, Python 3	centralized LocKedge accuracy (about 0.999) and time, at the Edge—1000 rounds, DoS-HTTP, DDoS-HTTP and theft-data give worse results than in centralized mode, others—better; edge computing capacity: maximum attack rate is 9600 samples per second, memory—up to 1800 samples per second	complexity of the algorithms, the training time performance, accuracy, DR, precision, recall, F1-score, CPU and RAM usage
[57]	hierarchical	NFBoT-IoT-v2	DDoS, DoS, Reconnaissance, Theft	balanced by the attacks/by the selected party	DFF	Custom, FedAvg	collaborating entities and non-collaborating entity scenario	accuracy, detection rate, FAR, F1-measure

Table 1. *Cont.*

Ref.	FL Architecture	Dataset	Attacks	Data Partition	ML Method	Implemen-Tation	Conducted Experiments and Best Accuracy	Used Metrics
[53]	centrilized, hyperledger fabric for privacy	KDD-Cup99	DoS, Probe, R2L, U2R	-	decision tree and multilayer perceptron—local model, FedAvg—global model training	custom	compare local model with random forest and SVM (50 iterations); compare the global model with CNN and DNN (100 iterations), the average AUC is 0.908	TP, FN, FP, TN, AUC, precision, recall, F1-score, time
[21]	centralized, Paillier public-key crypto-system-based secure communication protocol	gas pipelining system's dataset	DoS, DDoS, command injection, response injection, reconnaissance, command injection	even partitions	convolutional neural network and a gated recurrent unit	Keras API, Flask.2	local models, centralized mode, FL mode (99.20 for 7 agents, close to centralized mode results); from 2 to 10 rounds	accuracy, precision, recall, F-score
[24]	centralized (2 or 3 subnets)	KDD, NSLKDD, UNSW-NB15, N-BaIoT	anomalies, tested on DoS, R2L, U2R, probing attack, Mirai, BASHLITE	by subnets, not detailed	DNN to train local model, average gradient update algorithm for global model	Prototype, not detailed	for 2 and 3 subnets comparison with other methods; for 2 subnets the best ACC: KDD—97.52%, NSL-KDD—93.99%, UNSW—95.6%, N-BaIoT—99.84%; for 3 subnets: KDD—97.54%, NSL-KDD—93.37%, UNSW—95.67%, N-BaIoT—99.84%	time, privacy, TP, FP, TN, FN, ACC
[52]	centralized, secure gRPC channel	CSE-CIC-IDS2018, MQTTset, InSDN	DoS, DDoS, brute force, web-based, infiltration, botnet, Malformed, SlowITe, Flood, password guessing, probes, U2R	IID, Non-IID	deep neural networks, convolutional neural networks, recurrent neural networks	Google Colaboratory, Python 3, NumPy, Pandas, TensorFlow, Keras, Scikit-learn, and SMOTE libraries, Sherpa.ai FL framework	for the centralized model; FL scenario (accuracy close to the centralized): 5, 10, and 15 clients, 50 communication rounds	time complexity, power consumption, TP, FP, TN, FN, accuracy, precision, recall, F1-Score
[4]	centralized	SWAT 2015	signal injection	vertical partition	GBDT with Pailllier HE	FATE framework	2 scenarios: centralized scenario and FL scenario with 5 clients	accuracy, time complexity

5. Discussion and Conclusions

The analysis of the research papers allowed the authors to make following conclusions.

- The majority of the research papers are devoted to the design of network-based intrusion detection systems. The datasets that are used to evaluate the suggested approaches are represented either by PCAP packets or features extracted from them. Thus, while in [28] the authors used the ToN_IoT dataset, they do not consider the IoT telemetry data. Only in [52] is the MQTTset dataset that contains MQTT protocol-based communications between IoT devices used for the experiments. As it was stated in Section 3, this case corresponds to the horizontally partitioned data. The explanation of this fact is closely related to the current state of FL frameworks and libraries, the training time of the FL algorithms for vertically partitioned data, and the requirements to the computational and memory resources are extremely high. In [4], it was shown that inference time for decision trees in case of vertically partitioned data takes approximately 40 min.
- The proposed architecture for an IoT-based environment is typically centralized [20–22,28,52]. The clients are represented either by an IoT device directly or by a smart router that collects data from a set of IoT devices. The hierarchical architecture is presented in [23,57].
- The typical experimental scenario includes splitting one dataset across n clients in such a way that clients have datasets with different types of the attacks. In major cases, the class distribution in a client's dataset is balanced, and balancing dataset allows achieving higher results in intrusion detection efficiency. Only few works study the problem of the accuracy degradation due to imbalance in training data [22,28,52].
- The typical metrics that are used to evaluate the efficiency of the FL-based IDS systems are machine learning metrics that characterize the analysis model efficiency: accuracy, precision, recall, and F-measure. Only few research papers analyze the impact of the FL on the computational performance, impact on network traffic, and CPU load. Namely, in [52], the proposed FL-based algorithm is evaluated in terms of time and power consumption, and in [23], the complexity of the algorithms and the training time performance are considered. Thus, it is almost impossible to evaluate practical feasibility of the FL for intrusion detection.
- Scalability is often considered a certain advantage of FL; however, only few research papers focus on this problem [23,52,57], and the number of clients in the experiments rarely exceeds 10.
- A few works study the privacy issues in FL-based IDS, and propose novel algorithms with differential privacy mechanisms to increase the security of the data inputs. In this case, the authors focus on the evaluation of the analysis model efficiency, the issues relating to practical recommendations on how to select parameters of the differential privacy mechanisms in case the client's data imbalances are not discussed.

Thus, it is possible to conclude that, despite the certain benefits of FL for intrusion detection, such IDS systems are still a subject of the theoretical research. The research has shown that almost all practical challenges that are formulated at the beginning of the article are still open. Little effort has been spent on the analysis of the bias of the clients' data; different techniques are suggested for how to model it in the conditions of the lack of real-world datasets suitable for federated learning. In future research we plan to proceed with experimental evaluation of the federated learning application for the intrusion detection in the Internet of Things considering the outlined challenges, such as the bias of the clients' data.

Author Contributions: Conceptualization, E.F. and E.N.; methodology, E.N.; validation, E.F.; investigation, E.F., E.N. and A.S.; writing—original draft preparation, E.F. and A.S.; writing—review and editing, E.F. and E.N.; visualization, E.N.; project administration, E.N.; funding acquisition, E.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the grant of RSF #22-21-00724 in SPC RAS.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
DP	Differential Privacy
TEE	Trusted Execution Environment
FL	Federated Learning
ML	Machine Learning
IDS	Intrusion Detection System
NIDS	Network Intrusion Detection System
IoT	Internet of Things
GBDT	Gradient Boosting Decision Trees

References

1. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the AISTATS, Fort Lauderdale, FL, USA, 20–22 April 2017.
2. Lwakatare, L.E.; Raj, A.; Bosch, J.; Olsson, H.H.; Crnkovic, I. A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation. In Proceedings of the Agile Processes in Software Engineering and Extreme Programming, Montreal, QC, Canada, 21–25 May 2019; Kruchten, P., Fraser, S., Coallier, F., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 227–243.
3. Antonakakis, M.; April, T.; Bailey, M.; Bernhard, M.; Bursztein, E.; Cochran, J.; Durumeric, Z.; Halderman, J.A.; Invernizzi, L.; Kallitsis, M.; et al. Understanding the Mirai Botnet. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, Canada, 16–18 August 2017; USENIX Association: Vancouver, BC, Canada, 2017; pp. 1093–1110.
4. Novikova, E.; Doynikova, E.; Golubev, S. Federated Learning for Intrusion Detection in the Critical Infrastructures: Vertically Partitioned Data Use Case. *Algorithms* **2022**, *15*, 104. doi: 10.3390/a15040104. [CrossRef]
5. Ludwig, H.; Baracaldo, N.; Thomas, G.; Zhou, Y.; Anwar, A.; Rajamoni, S.; Ong, Y.J.; Radhakrishnan, J.K.; Verma, A.; Sinn, M.; et al. IBM Federated Learning: An Enterprise Framework White Paper V0.1. *arXiv* **2020**, arXiv:2007.10987.
6. Lo, S.K.; Lu, Q.; Zhu, L.; Paik, H.-Y.; Xu, X.; Wang, C. Architectural Patterns for the Design of Federated Learning Systems. *arXiv* **2021**, arXiv:2101.02373.
7. Ek, S.; Portet, F.; Lalanda, P.; Vega, G. A Federated Learning Aggregation Algorithm for Pervasive Computing: Evaluation and Comparison. In Proceedings of the 2021 IEEE International Conference on Pervasive Computing and Communications (PerCom), Kassel, Germany, 22–26 March 2021; pp. 1–10. [CrossRef]
8. Yurochkin, M.; Agarwal, M.; Ghosh, S.S.; Greenewald, K.H.; Hoang, T.N.; Khazaeni, Y. Bayesian Nonparametric Federated Learning of Neural Networks. In Proceedings of the ICML, Long Beach, CA, USA, 9–15 June 2019.
9. Mansour, A.B.; Carenini, G.; Duplessis, A.; Naccache, D. Federated Learning Aggregation: New Robust Algorithms with Guarantees. *arXiv* **2022**, arXiv:2205.10864.
10. Li, Q.; He, B.; Song, D. Model-Contrastive Federated Learning. *arXiv* **2021**, arXiv:2103.16257.
11. Lopez-Martin, M.; Sanchez-Esguevillas, A.; Arribas, J.I.; Carro, B. Supervised contrastive learning over prototype-label embeddings for network intrusion detection. *Inf. Fusion* **2022**, *79*, 200–228. [CrossRef]
12. Shahid, O.; Pouriyeh, S.; Parizi, R.M.; Sheng, Q.Z.; Srivastava, G.; Zhao, L. Communication Efficiency in Federated Learning: Achievements and Challenges. *arXiv* **2021**, arXiv:2107.10996.
13. Juvekar, C.; Vaikuntanathan, V.; Chandrakasan, A. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. In Proceedings of the 27th USENIX Conference on Security Symposium (SEC'18), Baltimore, MD, USA, 15–17 August 2018; USENIX Association: Boston, MA, USA, 2018; pp. 1651–1668.
14. Zhang, C.; Li, S.; Xia, J.; Wang, W.; Yan, F.; Liu, Y. BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning. In Proceedings of the 2020 USENIX Conference on Usenix Annual Technical Conference, Virtual Event, 15–17 July 2020; USENIX Association: Boston, MA, USA, 2020.
15. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawit, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and Open Problems in Federated Learning. In *Foundations and Trends in Machine Learning*; Now Publishers: Boston, MA, USA, 2021.
16. Truex, S.; Liu, L.; Chow, K.H.; Gursoy, M.E.; Wei, W. LDP-Fed: Federated learning with local differential privacy. In Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking, Heraklion, Greece, 27 April 2020.

17. Shokri, R.; Shmatikov, V. Privacy-preserving deep learning. In Proceedings of the 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 29 September–2 October 2015; pp. 909–910. [CrossRef]
18. Rieke, N.; Hancox, J.; Li, W.; Milletarì, F.; Roth, H.R.; Albarqouni, S.; Bakas, S.; Galtier, M.N.; Landman, B.A.; Maier-Hein, K.; et al. The future of digital health with federated learning. *NPJ Digit. Med.* **2020**, *3*, 119. [CrossRef]
19. Antunes, R.S.; André da Costa, C.; Küderle, A.; Yari, I.A.; Eskofier, B. Federated Learning for Healthcare: Systematic Review and Architecture Proposal. *ACM Trans. Intell. Syst. Technol.* **2022**, *13*, 1–23. [CrossRef]
20. Nguyen, T.D.; Marchal, S.; Miettinen, M.; Fereidooni, H.; Asokan, N.; Sadeghi, A.R. DÏoT: A Federated Self-learning Anomaly Detection System for IoT. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–9 July 2019; pp. 756–767.
21. Li, B.; Wu, Y.; Song, J.; Lu, R.; Li, T.; Zhao, L. DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber–Physical Systems. *IEEE Trans. Ind. Inform.* **2021**, *17*, 5615–5624. [CrossRef]
22. Rey, V.; Sánchez Sánchez, P.M.; Huertas Celdrán, A.; Bovet, G. Federated learning for malware detection in IoT devices. *Comput. Netw.* **2022**, *204*, 108693. [CrossRef]
23. Huong, T.T.; Bac, T.P.; Long, D.M.; Thang, B.D.; Binh, N.T.; Luong, T.D.; Phuc, T.K. LocKedge: Low-Complexity Cyberattack Detection in IoT Edge Computing. *IEEE Access* **2021**, *9*, 29696–29710. [CrossRef]
24. Khoa, T.V.; Saputra, Y.M.; Hoang, D.T.; Trung, N.L.; Nguyen, D.; Ha, N.V.; Dutkiewicz, E. Collaborative Learning Model for Cyberattack Detection Systems in IoT Industry 4.0. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea, 25–28 May 2020; pp. 1–6. [CrossRef]
25. Long, G.; Tan, Y.; Jiang, J.; Zhang, C. Federated Learning for Open Banking. *arXiv* **2020**, arXiv:2108.10749.
26. Ahmed, U.; Srivastava, G.; Lin, J.C.W. Reliable customer analysis using federated learning and exploring deep-attention edge intelligence. *Future Gener. Comput. Syst.* **2022**, *127*, 70–79. [CrossRef]
27. Li, J.; Cui, T.; Yang, K.; Yuan, R.; He, L.; Li, M. Demand Forecasting of E-Commerce Enterprises Based on Horizontal Federated Learning from the Perspective of Sustainable Development. *Sustainability* **2021**, *13*, 13050. [CrossRef]
28. Campos, E.M.; Saura, P.F.; González-Vidal, A.; Hernández-Ramos, J.L.; Bernabé, J.B.; Baldini, G.; Skarmeta, A. Evaluating Federated Learning for intrusion detection in Internet of Things: Review and challenges. *Comput. Netw.* **2022**, *203*, 108661. [CrossRef]
29. Novikova, E.; Fomichov, D.; Kholod, I.; Filippov, E. Analysis of Privacy-Enhancing Technologies in Open-Source Federated Learning Frameworks for Driver Activity Recognition. *Sensors* **2022**, *22*, 2983. [CrossRef]
30. Lyu, L.; Yu, H.; Yang, Q. Threats to Federated Learning: A Survey. *arXiv* **2020**, arXiv:2003.02133.
31. Agrawal, S.; Sarkar, S.; Aouedi, O.; Yenduri, G.; Piamrat, K.; Bhattacharya, S.; Maddikunta, P.K.R.; Gadekallu, T.R. Federated Learning for Intrusion Detection System: Concepts, Challenges and Future Directions. *arXiv* **2021**, arXiv:2106.09527.
32. Bellatreche, L.; Boukhalfa, K.; Richard, P. Data Partitioning in Data Warehouses: Hardness Study, Heuristics and ORACLE Validation. In Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK '08), Turin, Italy, 2–5 September 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 87–96. [CrossRef]
33. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17), Dallas, TX, USA, 30 October–3 November 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 1175–1191. [CrossRef]
34. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecur* **2019**, *2*, 20. [CrossRef]
35. Bukhanov, D.G.; Polyakov, V.M. Detection of network attacks based on adaptive resonance theory. *J. Phys. Conf. Ser.* **2018**, *1015*, 042007. [CrossRef]
36. Yunwu, W. Using Fuzzy Expert System Based on Genetic Algorithms for Intrusion Detection System. In Proceedings of the 2009 International Forum on Information Technology and Applications, Chengdu, China, 15–17 May 2009; Volume 2, pp. 221–224. [CrossRef]
37. Dave, M.H.; Sharma, S.D. Improved Algorithm for Intrusion Detection Using Genetic Algorithm and SNORT. *Int. J. Emerg. Technol. Adv. Eng.* **2014**, *4*, 273–276.
38. Ranjan, R.; Sahoo, G. A New Clustering Approach for Anomaly Intrusion Detection. *Int. J. Data Min. Knowl. Manag. Process. (IJDKP)* **2014**, *4*, 29–38. [CrossRef]
39. Li, Z.; Qin, Z.; Huang, K.; Yang, X.; Ye, S. Intrusion Detection Using Convolutional Neural Networks for Representation Learning. In Proceedings of the International Conference on Neural Information Processing (ICONIP), Guangzhou, China, 14–18 November 2017; Springer: Cham, Switzerland, 2017; Volume 10638. [CrossRef]
40. Jianwei, H.; Chenshuo, L.; Yanpeng, C. An Improved CNN Approach for Network Intrusion Detection System. *Int. J. Netw. Secur.* **2021**, *23*, 569–575. [CrossRef]
41. Vinayakumar, R.; Soman, K.; Poornachandran, P. Evaluation of Recurrent Neural Network and Its Variants for Intrusion Detection System IDS. *Int. J. Inf. Syst. Model. Des.* **2017**, *8*, 43–63. [CrossRef]
42. Song, Y.; Hyun, S.; Cheong, Y.G. Analysis of Autoencoders for Network Intrusion Detection. *Sensors* **2021**, *21*, 4294. [CrossRef] [PubMed]

43. Gajewski, M.; Batalla, J.M.; Mastorakis, G.; Mavromoustakis, C.X. A distributed IDS architecture model for Smart Home systems. *Clust. Comput.* **2017**, *22*, 1739–1749. [CrossRef]
44. Shterenberg, S.I.; Poltavtseva, M.A. A Distributed Intrusion Detection System with Protection from an Internal Intruder. *Autom. Control Comput. Sci.* **2018**, *52*, 945–953. [CrossRef]
45. Schueller, Q.; Basu, K.; Younas, M.; Patel, M.; Ball, F. A Hierarchical Intrusion Detection System using Support Vector Machine for SDN Network in Cloud Data Center. In Proceedings of the 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), Sydney, Australia, 21–23 November 2018; pp. 1–6. [CrossRef]
46. Saghezchi, F.B.; Mantas, G.; Ribeiro, J.; Al-Rawi, M.; Mumtaz, S.; Rodriguez, J. Towards a secure network architecture for smart grids in 5G era. In Proceedings of the 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain, 26–30 June 2017; pp. 121–126. [CrossRef]
47. Zhang, Y.; Wang, L.; Sun, W.; Green, R.C., II; Alam, M. Distributed Intrusion Detection System in a Multi-Layer Network Architecture of Smart Grids. *IEEE Trans. Smart Grid* **2011**, *2*, 796–808. [CrossRef]
48. Javed, Y.; Felemban, M.; Shawly, T.; Kobes, J.; Ghafoor, A. A Partition-Driven Integrated Security Architecture for Cyberphysical Systems. *Computer* **2020**, *53*, 47–56. [CrossRef]
49. Kholod, I.; Yanaki, E.; Fomichev, D.; Shalugin, E.; Novikova, E.; Filippov, E.; Nordlund, M. Open-Source Federated Learning Frameworks for IoT: A Comparative Review and Analysis. *Sensors* **2021**, *21*, 167. [CrossRef] [PubMed]
50. Zhang, Z.; Zhang, Y.; Guo, D.; Yao, L.; Li, Z. SecFedNIDS: Robust defense for poisoning attack against federated learning-based network intrusion detection system. *Future Gener. Comput. Syst.* **2022**, *134*, 154–169. [CrossRef]
51. Ibitoye, O.; Shafiq, M.O.; Matrawy, A. Differentially private self-normalizing neural networks for adversarial robustness in federated learning. *Comput. Secur.* **2022**, *116*, 102631. [CrossRef]
52. Friha, O.; Ferrag, M.A.; Shu, L.; Maglaras, L.; Choo, K.K.R.; Nafaa, M. FELIDS: Federated learning-based intrusion detection system for agricultural Internet of Things. *J. Parallel Distrib. Comput.* **2022**, *165*, 17–31. [CrossRef]
53. Hei, X.; Yin, X.; Wang, Y.; Ren, J.; Zhu, L. A trusted feature aggregator federated learning for distributed malicious attack detection. *Comput. Secur.* **2020**, *99*, 102033. [CrossRef]
54. Zhao, R.; Yin, Y.; Shi, Y.; Xue, Z. Intelligent intrusion detection based on federated learning aided long short-term memory. *Phys. Commun.* **2020**, *42*, 101157. [CrossRef]
55. Kumar, K.S.; Nair, S.A.H.; Guha Roy, D.; Rajalingam, B.; Kumar, R.S. Security and privacy-aware Artificial Intrusion Detection System using Federated Machine Learning. *Comput. Electr. Eng.* **2021**, *96*, 107440. [CrossRef]
56. Astillo, P.V.; Duguma, D.G.; Park, H.; Kim, J.; Kim, B.; You, I. Federated intelligence of anomaly detection agent in IoTMD-enabled Diabetes Management Control System. *Future Gener. Comput. Syst.* **2022**, *128*, 395–405. [CrossRef]
57. Sarhan, M.; Lo, W.W.; Layeghy, S.; Portmann, M. HBFL: A Hierarchical Blockchain-based Federated Learning Framework for a Collaborative IoT Intrusion Detection. *arXiv* **2022**, arXiv:2204.04254.
58. Dankmar, B. Multinomial logistic regression algorithm. *Ann. Inst. Stat. Math.* **1992**, *44*, 197–200.
59. Li, X.; Huang, K.; Yang, W.; Wang, S.; Zhang, Z. On the Convergence of FedAvg on Non-IID Data. In Proceedings of the 8th International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
60. Yu, P.; Wynter, L.; Lim, S.H. Fed+: A Family of Fusion Algorithms for Federated Learning. *arXiv* **2020**, arXiv:2009.06303.
61. Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access* **2020**, *8*, 165130–165150. . [CrossRef]
62. Evaluating-FL-for-Intrusion-Detection-in-IoT-Review-and-Challenges Datasets (2021). Available online: https://github.com/Enrique-Marmol/Evaluating-FL-for-Intrusion-Detection-in-IoT-review-and-challenges (accessed on 15 March 2022).
63. A Micro Web Framework Written in Python. Available online: https://flask.palletsprojects.com/en/2.1.x/ (accessed on 15 March 2022).
64. Flask Socketio. Available online: https://flask-socketio.readthedocs.io/en/latest/ (accessed on 15 March 2022).
65. Flask Socketio Client. Available online: https://github.com/socketio/socket.io-client (accessed on 15 March 2022).
66. Gevent Asynchronous Framework. Available online: https://github.com/gevent/gevent (accessed on 15 March 2022).
67. Keras Deep Learning Library. Available online: https://faroit.github.io/keras-docs/2.0.2/ (accessed on 15 March 2022).
68. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. doi: 10.1109/MPRV.2018.03367731. [CrossRef]
69. Fed_IoT_Guard. 2021. Available online: https://github.com/ValerianRey/fed_iot_guard (accessed on 30 March 2022).
70. Moustafa, N. The Bot-IoT Dataset. 2019. Available online: https://research.unsw.edu.au/projects/bot-iot-dataset (accessed on 13 July 2022).
71. Chai, H.; Leng, S.; Chen, Y.; Zhang, K. A Hierarchical Blockchain-Enabled Federated Learning Algorithm for Knowledge Sharing in Internet of Vehicles. *Trans. Intell. Transport. Sys.* **2021**, *22*, 3975–3986. [CrossRef]
72. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
73. Keras: Python Deep Learning Library. Available online: http://keras.io/ (accessed on 26 March 2022).
74. Morris, T.; Gao, W. Industrial Control System Traffic Datasets for Intrusion Detection Research. In Proceedings of the International Conference on Critical Infrastructure Protection, Arlington, VA, USA, 17–19 March 2014; Volume 441, pp. 65–78. [CrossRef]

75. Schneble, W.; Thamilarasu, G. Attack Detection Using Federated Learning in Medical Cyber-Physical Systems. In Proceedings of the 28th International Conference on Computer Communications and Networks, Valencia, Spain, 29 July–1 August 2019.
76. Chen, Y.; Qin, X.; Wang, J.; Yu, C.; Gao, W. FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare. *IEEE Intell. Syst.* **2020**, *35*, 83–93. [CrossRef]
77. KDD Dataset. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 15 March 2022).
78. University of New Brunswick Dataset. Available online: https://www.unb.ca/cic/datasets/nsl.html (accessed on 15 March 2022).
79. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive dataset for network intrusion detection systems (UNSW-NB15 network dataset). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6. [CrossRef]
80. FATE. An Industrial Grade Federated Learning Framework. Available online: https://fate.fedai.org/ (accessed on 25 June 2022).
81. Secure Water Treatment (SWaT). Available online: https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/(accessed on 25 June 2022).
82. Google Colaboratory. 2021. Available online: https://colab.research.google.com/ (accessed on 30 March 2022).
83. Rodríguez-Barroso, N.; Stipcich, G.; Jiménez-López, D.; Ruiz-Millán, J.A.; Martínez-Cámara, E.; González-Seco, G.; Luzón, M.V.; Veganzones, M.A.; Herrera, F. Federated Learning and Differential Privacy: Software tools analysis, the Sherpa.ai FL framework and methodological guidelines for preserving data privacy. *Inf. Fusion* **2020**, *64*, 270–292. [CrossRef]
84. Anthony, L.F.W.; Kanding, B.; Selvan, R. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv* **2020**, arXiv:2007.03051.