Privacy in Federated Learning Models for Intrusion Detection Systems

ERIK SCHNEIDER, Department of Computer Science, Colorado State University, Fort Collins, Colorado, USA
BRUHADESHWAR BEZAWADA, Mathematics and Computer Science, Southern Arkansas University, Magnolia, Arkansas, USA
INDRAKSHI RAY, Department of Computer Science, Colorado State University, Fort Collins, Colorado, USA

Federated learning-based solutions enable multiple organizations to build a machine learning model using the collective knowledge of the organizations without exchanging the sensitive raw data of any participant. The key roadblock for widespread adoption is the possibility of training data leakage in federated learning models, which have been reported in federated learning models for image processing, natural language processing, and tabular data. However, the impact and feasibility of such privacy leakage attacks on federated learning-based intrusion detection systems (IDS) remains largely unexplored.

In this work, for the first time, we focus on the problem of training data privacy in federated learning-based network IDS by exploring the implementations, techniques, and limitations of existing federated IDS solutions. First, we provide a systematic study of the landscape of machine learning-based network IDS and federated learning-based network IDS. Second, we demonstrate that data leakage attacks on federated learning-based IDS are yet to be explored to the full extent by the research community. Finally, we describe the existing defenses against such attacks and suggest the possibility of using these approaches for federated network IDS in the future.

CCS Concepts: • **Security and privacy** → **Intrusion detection systems**; **Privacy protections**; • **Computing methodologies** → *Neural networks*;

Additional Key Words and Phrases: Intrusion detection systems, Privacy preserving, Deep learning, Federated learning, Threat intelligence

ACM Reference format:
Erik Schneider, Bruhadeshwar Bezawada, and Indrakshi Ray. 2026. Privacy in Federated Learning Models for Intrusion Detection Systems. *Digit. Threat. Res. Pract.* 7, 3, Article 17 (August 2026), 25 pages.
https://doi.org/10.1145/3828661

1 Introduction

Network threats are increasing at a rapid rate [59] each year, and as a result, organizations are suffering severe financial losses and operational disruptions [24]. **Intrusion Detection Systems (IDS)**, which use signature-based, anomaly-based, and of late, machine learning-based techniques, have proven to be effective first line of defenses. Due to the rapid advancements in AI, there has been remarkable progress in machine learning-based

This work was supported in part by National Institute of Standards and Technology award number 60NANB23D152 and by Division of Computer and Network Systems award number CNS 2335687.
Authors' Contact Information: Erik Schneider, Department of Computer Science, Colorado State University, Fort Collins, Colorado, USA; e-mail: e.schneider@cannyit.de; Bruhadeshwar Bezawada (corresponding author), Mathematics and Computer Science, Southern Arkansas University, Magnolia, Arkansas, USA; e-mail: bez.bru@gmail.com; Indrakshi Ray, Department of Computer Science, Colorado State University, Fort Collins, Colorado, USA; e-mail: Indrakshi.Ray@colostate.edu.

ACM 2576-5337/2026/8-ART17
https://doi.org/10.1145/3828661

IDS [12, 26, 60, 63, 76, 80, 94] and they are well positioned to become the *de-facto* network defenses in the future. The use of machine learning for security is likely to increase in the future as the attacks become more complex and generate more data, which require deeper analysis and real-time decision making.

Machine learning IDS solutions are very effective against recognized network threats, and variants of known attacks, but are generally ineffective against unseen attacks. To protect against emerging and unseen threats, it is critical and necessary for organizations to be willing to share their network attack intelligence with their peers on a regular basis. But data confidentiality is a major roadblock as network intelligence sharing involves revealing the detailed attack data [33], such as the systems affected and services disrupted, which are confidential to an organization and its operations. These critical requirements have led to the development of privacy-preserving threat intelligence sharing techniques and platforms.

Privacy-Preserving Threat Intelligence Sharing. Several approaches have been described to address the privacy-preserving sharing of network threat intelligence. Researchers have proposed cryptographic methods to solve the problem of secure data sharing. These include homomorphic encryption [1], data anonymization [8], **Differential Privacy (DP)** [19, 99], secure multi-party computation [52, 102], and privacy-preserving machine learning [58, 62, 68, 75]. However, these techniques impose significant setup overhead, introduce key management, and scalability challenges. The **National Institute of Standards and Technology (NIST)** describes the guidelines for sharing threat intelligence information in `NIST SP 800-150` [36]. **Personally Identifiable Information (PII)**, Controlled Unclassified Information, government classified information, and company proprietary information can be leaked through network threat intelligence sharing. Moreover, leaking such sensitive information without consent may impact laws, such as, including General Data Protection Regulation, and Gramm-Leach-Bliley Act. The European Network and Information Security Agency published documents [16, 22] in an attempt to collate the various forms of Cyber Threat Intelligence sharing and summarize the various alert formats and standardization efforts.

***Federated Learning (FL)** and Shared IDS.* FL [58], where each party trains on its own data and shares only the trained neural network parameters, can be used to provide a more robust model that can detect various types of attacks [41]. However, the confidentiality claims of FL methods have been challenged by targeted attacks [107] that have successfully extracted training data in applications like image processing and tabular data. While there are no reported attacks against **FL-Based IDS (FIDS)**, there is need for deeper analysis and research on this aspect.

In our work, for the first time, we focus on the confidentiality of training data of FIDS and provide a deep dive into the possibility and severity of such attacks. We begin by providing a systematic study of the landscape of machine learning-based network IDS and FL-based network IDS. We clearly highlight the differences among the various techniques such as privacy preserving machine learning, collaborative IDS, and federated IDS. Next, we demonstrate that data leakage attacks on FL-based IDS are yet to be explored to their full extent by the research community and we describe a systems engineering approach to characterize and quantify the possible attacks. We analyze the possible attacks and respective defenses using the IEC 60812 **Failure Modes Effects and Criticality Analysis (FMECA)** standard, we provide a quantifiable method for understanding the confidentiality threats and the effectiveness of the defenses against such threats. Finally, based on existing FL defense frameworks, we describe possible defenses against such attacks in various data domains and suggest that some of these approaches may be suitable for federated network IDS in the future.

Organization. The rest of the article is organized as follows. In Section 2, we provide a technical overview of machine learning-based network IDS. In Section 3, we describe the attempts to share intelligence using privacy-preserving machine learning techniques. In Section 4, we describe existing FIDS methods and a brief systematization of the research in this area by comparing them based on the attacks possible, the defenses used, and the general challenges faced by the respective techniques. In Section 5, we describe our threat analysis approach for the attacks against FIDS using the FMECA approach as well as some general data privacy attacks that are possible on FIDS like data leakage during transfer and their expected severity. In Section 6, we describe

the targeted attacks against FIDS and their severity. In Section 7, we summarize the possible defenses and evaluate their effectiveness for FIDS using the FMECA method. In Section 8, we provide concluding remarks on privacy attacks on FIDS and highlight the future work possible in this space. Appendix A (Table A1) lists expansions of some important acronyms used in the article.

2 Machine Learning-Based Network IDS

In this section, we present a systematic study of machine learning-based network IDS. First, we describe our data collection methodology in selecting the appropriate papers to include. Second, we describe centralized machine learning-based IDS, and finally, we discuss **Collaborative IDS (CIDS)**.

2.1 Data Collection Methodology

We reviewed publications on the specific topics of FL and intrusion detection, as well as potential leaks and attacks on **Deep Neural Networks (DNNs)**, and the challenges of collaborating and sharing network attack data. Only papers that had recently been published in top peer-reviewed conferences and journals were taken into account. The keywords used for this process include "machine learning based intrusion detection systems," "machine learning based IDS," "machine learning based network intrusion detection systems," "federated learning based intrusion detection systems," "collaborative intrusion detection systems," "sharing network attack data," "sharing threat intelligence," "attacks on federated learning based intrusion detection systems," "attacks on machine learning based intrusion detection systems," "survey of federated learning models," and "survey of machine learning based intrusion detection models." Databases including Google scholar, IEEE Explore, Elsevier, and Springer were considered.

We then examined the identified publications to determine the extent to which they address the attacks, defenses, and challenges we identified. The keywords used for this process include "defenses against training data leakage," "attacks and defenses on federated learning," "attacks and defenses on privacy-preserving deep learning," "defenses against deep gradient leakage," "defenses against gradient-inversion," "adversarial attacks on federated learning," "training data leakage in deep learning models," "survey of attacks on federated learning models," and "survey of defenses for federated learning models." A general finding of our analysis is that the topic of FL is becoming increasingly prominent. Its use in the field of network security and threat intelligence is also being increasingly researched, but real-world applications using FL are lacking. The topic we examined, namely the threat to privacy in the context of FIDS, is only being actively addressed by a small community of researchers.

Machine learning techniques for intrusion detection thrive on the premise that attack data exhibits familiar patterns, and by learning the previous patterns, one may identify new attacks that exhibit similar patterns. The key goal of these techniques is to train a machine learning model to (a) distinguish between benign traffic and malicious traffic and/or (b) distinguish among multiple types of attacks.

Machine learning-based network IDS can be classified broadly into two categories based on their training model and decision-making approach. First, in centralized machine learning techniques the training data is available with a single centralized entity who is in charge of training the machine learning model and deploying the intrusion detection system. Second, in collaborative techniques multiple parties might engage in both the training process and the decision making process. We will provide a detailed technical overview of both these categories in the following sections.

2.2 Centralized Machine Learning-Based IDS

Under the centralized categorization, several prominently recognized solutions [12, 26, 60, 63, 76, 94] have proposed a multitude of excellent machine learning-based techniques, primarily for anomaly detection. These solutions may be differentiated into (a) closed-world and (b) open-world solutions. Loosely speaking, a "closed-world" solution is a machine learning solution that exhibits the following solution template.

(1) The data necessary to train the machine learning model is collected by security experts over a period of time and are labeled appropriately. Clearly, this process is cumbersome, expensive, and requires the availability of expert personnel on the premises.
(2) The next phase is to design a suitable machine learning model and train the model following best practices. The best practices include prevention of bias in data processing, avoiding over-fitting, and parameter tuning using various optimization techniques.
(3) The trained model is then deployed on a suitable computing platform that monitors the incoming and outgoing traffic. The configuration of such a platform depends on the nature of machine learning model, the required traffic flow rates, and the volume of attacks that may be possible.

Therefore, the closed-world framework solutions are subject to several constraints, and their deployment requires the utmost care. The limitations of the closed-world centralized IDS were exposed by Sommer and Paxson [80] and Mirsky et al. [60], who showed that centralized IDS were already prevalent in academic and industry circles.

Besides the different data sources and quality, a typical and challenging open world requirement is to transform IDS into intrusion prevention systems. This means not only detecting possible attacks or anomalies in a dataset, but also deciding whether a packet or flow is to be blocked immediately to prevent an attack. Compared to backward post-communication analysis, this is much more challenging. Unfortunately, the data and evaluations in many publications are not comparable, making benchmarking difficult. This makes it all the more difficult to assess when or whether a model is capable of correctly classifying incoming packets. However, this is crucial for genuine open world use in reality.

Sommer and Paxson [80] provide analytical and qualitative arguments regarding this premise and how it has hindered the real-world deployment of network intrusion detection. This work is a landmark report on the development and effectiveness of IDS in the real-world. They highlight several important facets of intrusion detection including high cost of errors, lack of training data, semantic gap between results and their operational interpretation. One of the key aspects they mention is the lack of real data in evaluating an IDS that is built in a laboratory setting, which is termed as "closed-world" settings. Such a system should be tested on multiple datasets that fall outside the local context of the laboratory setting and as a result is more likely to adapt to different attacks and variations of attacks. Note that, by definition and design, these systems are naturally unsuitable for detecting new or unknown attacks that exhibit little or no similarity with currently known attack patterns. Furthermore, the real-time effectiveness of the machine learning model is of paramount importance. Similar to product reviews on commercial goods and services, there is a significant need for critical feedback on the performance of such systems from the actual users of the system. The feedback should be used to update and improve the model while focusing specifically on the weaker aspects of the model.

Recognizing the limitations of *closed-world* solutions and the need for *open-world*, the KitNET framework [60] describes an open-world oriented solutions to overcome some of the limitations of closed-world solutions. One important aspect highlighted by the training method in the KitNET framework [60] is that of single instance training being preferable to batch training for open-world online settings. The batch training is done using the gradient descent optimization [57] where the errors in training are averaged and back-propagated to update the model weights; this approach requires more training instances, a batch sized set, to be stored during the training phase. Whereas the single-instance training is done using **Stochastic Gradient Descent (SGD)** [23] optimization and the errors are used to update the model weights for each instance and therefore requires less memory. In summary, closing the gap between closed-world and open-world solutions is an ongoing effort by the research community.

Recently, Apruzzese et al. [4] described a promising approach to bridge this gap effectively. In comparison to the standard KDD99 [86, 90], the availability of additional ground-truth labeled datasets [25, 72, 74] has contributed toward the progress in developing more effective evaluation methods for IDS. Apruzzese et al. [4] describe a *cross-evaluation* of machine learning-based IDSs given such datasets where an IDS is evaluated against other types

of datasets in a data agnostic manner, which represents another significant step toward open-world oriented IDS solutions.

2.3 CIDS

The community has recognized the need to develop and deploy an IDS using multiple data sources, which have led to the evolution of CIDS. The term "Collaborative Intrusion Detection System" (CIDS) has been traditionally used to describe a collection of multiple monitoring stations and analysis units that work together to detect intrusions. The group of monitoring stations provide alerts and data to the analysis units where an analyst or collection of analysts perform an integrated analysis of the data and alerts. The motivation for CIDS is the need for scaling the IDS as per the size of the network and the scale of the attacks, both of which are constantly increasing. There are three types of CIDS [91] possible based on the communication topologies among the analysts and the monitors along with the number of analysts used. In centralized CIDS, multiple monitors provide alerts and data to a single analyst. In hierarchical CIDS, the monitors and the analysts are arranged in certain hierarchy where the data flows from the monitor hierarchy to the analyst hierarchy. In distributed CIDS, the monitors also combine with analyst functionality and share their analysis reports and data with each other.

The use of CIDS is fairly common in modern networks with the presence of technologies like Carbon Black [11] and Crowdstrike [7], where each host or networking entry point is monitored for potential alerts. A dedicated local server or a group of cloud servers perform the task of analyzing the data for intrusions. The most popular CIDS are centralized and hierarchical due to their ease of administration and control. At the time of writing, to the best of our knowledge, we are not aware of a distributed CIDS being deployed in practice.

In CIDS, the key aspect is that the sensory data or offending network packets are shared with a central monitor as there is trust relationship or a leader/follower relationship among the monitor-analyst entities. Therefore, despite the apparent practical benefits of CIDS, this model is unlikely to go beyond a single enterprise due to valid data privacy violation concerns. As a result, the research community has focused their efforts on the development of privacy-preserving techniques for sharing network threat intelligence signals with other interested parties without revealing the raw traffic sources that generate the signals and these techniques are described in the following section.

3 Privacy-Preserving Machine Learning Models

While CIDS are powerful, they are limited to a single organization as organizations are at cross-roads when it comes to trading off data privacy for better machine learning model performance. To address this important problem, two varying types of decentralized privacy preserving learning models [58, 75] have been proposed: (a) distributed deep learning models and (b) FL models. In this section, we give an overview of these two models, which are quite similar in spirit and exhibit a few technical differences during the training process.

3.1 Distributed Deep Learning

Shokri et al. [75] described a distributed training technique, based on selective SGD learning method, which requires the exchange of gradients with other parties. Initially, the collaborative training framework assumes that there are N-participants in the training process, although some parties may decide to drop off or new parties may get added to the training process. A centralized parameter server is responsible for managing a list of global parameters, which will ultimately be the common model by inputs from all the local clients. Initially, each participant connects to the global parameter server and receives the structure of a neural network model to be trained. Each client maintains a list of local parameters, i.e., weights and biases, which can be initialized at random. A gradient is a value by which a particular weight needs to be adjusted to reduce the error rate of the learning model and improve the performance. Now, each participant trains the neural network on its own private data using the SGD [44, 73, 88] method or the variants thereof [70, 89]. Periodically, the local parties send their

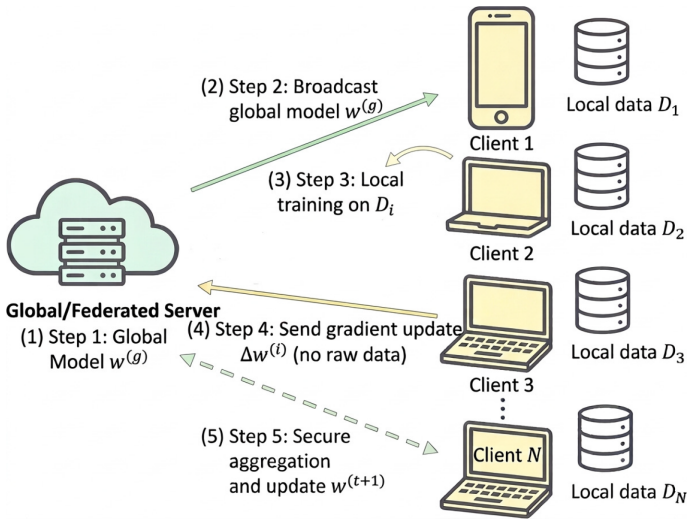

Fig. 1. Distributed deep learning/FL at round t. Aggregated weight updates after round t are w^{t+1}.

most significant gradients to the parameter server for dissemination to the remaining participants. The server performs aggregation-based inputs received from one or more clients and disseminates the updated weights to the clients. The other parties can selectively update their respective local weights, i.e., the weights with the same indexes as the gradients, thereby, adjusting the weight to a common value across all the participants. We provide more technical details in the following discussion.

The model training is carried out by N parties, each referred to as a local client or simply a participant. A central parameter server stores and updates the global parameters, constituting the joint model learned collaboratively by all local clients. All clients and the central parameter server agree on a shared neural network architecture. The client keeps a set of local parameters—namely the weight gradients and bias gradients—which it may either initialize or keep unchanged. For participant i, this set of parameters is denoted by $w^{(i)}$.

Once initialization is complete, each participant begins local training by optimizing its neural network on its own private dataset using an algorithm such as SGD [23, 44]. This local training proceeds over multiple epochs until a predefined stopping criterion is satisfied. During this phase, each participant trains independently of the others. They do not modify one each other's models directly; any interaction occurs only indirectly through the shared parameter server.

The algorithmic steps of the local training procedure executed on each participant's side are outlined below.

(1) Set the local model parameters to $w^{(g)}$, as received from the server, and the learning rate as α. This corresponds to Step 2 in Figure 1.
(2) Iterate until either the predefined number of epochs is reached or the target minimum error criterion is satisfied. This corresponds to Step 3 in Figure 1.
   (a) After each epoch, the gradient vector, $\Delta w^{(i)}$, is computed based as: $\Delta w^{(i)} = w_j - w_i$, where w_j are the weights after training and w_i are the weights before the training, for all the weights in the neural network.
   (b) For a specified threshold Θ_u, the client selects and uploads the Θ_u most significant gradients to the server, say, the values that are the largest. This corresponds to Step 4 in Figure 1.

(c) Once the server signals a processing of all the clients gradient updates, each client downloads the updated neural parameters from the server and replaces the respective local parameters in the client's model. This corresponds to Step 5 in Figure 1.
(d) Run the next epoch of training on the model with the updated parameters on the next batch of local data and repeat the gradient transmission process.

The central server manages the global parameter vector Δw^g, of the common global model, using the process shown below.

(1) The server initializes the global parameters $w^{(g)}$ to random values. This corresponds to Step 1 in Figure 1.
(2) The server broadcasts the initial global parameters $w^{(g)}$ to all the clients. This corresponds to Step 2 in Figure 1.
(3) When a participant uploads the weight-gradients $w^{(i)}$: The server updates the global parameters by adding/subtracting the corresponding values of $w^{(i)}$ and $w^{(g)}$ as: $\Delta w^g = \Delta w^g \pm \Delta w_j$, where the value of uploaded gradient for jth parameter is Δw^g and the addition/subtraction depends on the sign of the respective gradient in the vector. This corresponds to Step 5 in Figure 1.
(4) Finally, the server sends all the parameters in the vector $w^{(g)}$ to the participant. This corresponds to Step 5 in Figure 1.

This method ensures that at the end of the training process, all the participants share similar weight vectors more or less. More importantly, the training data does not leave the premises of the local participants providing the necessary data privacy.

3.2 FL

FL [9, 58, 83, 99] is another decentralized approach for training a machine learning model collaboratively. The key difference is that the global parameter server performs an averaging operation over all the received gradients and disseminates the updated weights to the participants. The weights are adjusted according to the product of the learning rate and the weighted average of the changed gradients. There are two possible variants of this step. First, in classical FedAVG, clients usually send model weights or weight updates, not raw gradient vectors. The server aggregates those client updates by averaging them, then broadcasts the new global model back to the clients. Second, in a SGD variant called FedSGD, each client computes a gradient on its local data and sends the gradient vector to the server, which averages gradients directly. The server aggregates gradients and applies the update centrally and updates the clients as before. This step is logically equivalent to the server computing the average of the local models of the clients. FedSGD method has been proven to adjust to unbalanced and non-IID data distributions. Also, the number of rounds of communication is significantly smaller as the central server performs the periodic updates.

Distributed Deep Learning vs. FL. We note that the choice of either methods discussed in this section can depend on various operational factors. Both these methods achieve similar results with varying costs of computation and communication for the clients and the server. All the attacks and defenses apply to both methods for all practical purposes.

4 FL-Based IDS

First, we provide the technical perspectives in FIDS, i.e., the distinction among the implementations possible in FIDS. Then, we provide a brief state-of-the-art "Systematization-of-Knowledge" of FIDS.

4.1 FIDS in Trusted and Untrusted Environments

FIDS solutions explicitly focus on the use of machine learning for IDS and emphasize on learning from other collaborative parties and have garnered significant interest from the research community [14, 20, 43, 48]. These

systems have attempted to solve several problems such as lack of data diversity, detecting unknown attacks, and reducing impact of data poisoning. We highlight two modalities of FIDS, which put forward two distinct case studies of trust and data privacy:

—Trusted environments: In CIDS, all sensors and decision makers are part of the same organization and there is complete trust among the entities. A CIDS can trivially operate as FIDS by having one agent perform the federated aggregation and enabling the decision makers to use the FL model. This implementation mechanism has often been captured under the term "Edge computing" [81] where the individual decision makers may not have the computational and storage resources suitable for building an effective IDS. A use case here is IDS at the edge routers of a large network, which are designed to protect all relevant transfer and entry points from attacks. Identified attacks shall also be identified and intercepted at all other transfer points. Since all IDSs are controlled and operated by one organization and the environment is trusted, data leakage among participants is not a concern.

—Untrusted environments: The original intention of FIDS has been to create an IDS based on cross-domain knowledge, i.e., to use the knowledge of other domains or parties to build a highly effective IDS. In this context, one may distinguish two models here: distributed and centralized. A "centralized" FIDS follows the spirit of FL [58] where a global server aggregates the weights of the individual parties using the FedAVG algorithm and disseminates the final weights to the participants. A "distributed" FIDS follows a peer-to-peer distributed learning model [75] where the participants periodically exchange the gradients and perform local aggregation based on their own policies or algorithms. We wish to note that, a majority of the FIDS follow the centralized model due to the simplicity and uniformity of the federated averaging algorithm, and the relative ease of implementation. The cross-domain FIDS requires establishing trust relationships and other privacy protection controls among the participants. While FL is specifically intended to protect data privacy, the adversarial behavior of one or more malicious participants must be accounted for and proper defenses should be envisaged by the other honest participants. This notion has been formalized and studied more rigorously [21, 79] in recent times under the Byzantine failure model [46, 54, 55]. The Byzantine failure model seeks solutions that work even when $\lfloor \frac{n-1}{3} \rfloor$ participants out of the n total participants are malicious. This fresh perspective creates new avenues for more effective solutions and requires further research. The use case involves IDS operated by organizations that do not trust or even know each other. These organizations want to share attack data to identify and defeat attacks on other participants' networks. Such an IDS shall also be able to deal with malicious members, and the confidentiality of the data is paramount.

4.2 A Brief Systematization of Knowledge of FIDS

Table 1 offers a comprehensive summary of the publications reviewed and establishes a taxonomy of the identified attacks, defenses, and challenges. In the table, the publications were initially categorized according to the attacks discussed, the defense mechanisms, and the challenges. Subsequent to this initial classification, the subjects were then further categorized into the subcategories that were the focus of the examination. For each publication, a classification has been assigned according to the extent to which it addresses the relevant topic (indicated by $\surd$ in the table), does not address it (indicated by $\times$ in the table), or is not relevant at all in the context of the publication (indicated by - in the table).

A more detailed discussion of these respective publications is provided in the relevant sections. As anticipated, it has become apparent that the publications either address specific subjects or, as in the case of surveys, encompass a broad range of issues. However, a thorough review of the evaluated publications reveals that none of them address all of the points identified. The most frequently discussed topics are the influences of model architecture and the challenges posed by skewed and biased data, as well as FL. A paucity of publications has examined the challenges of threat intelligence in this context and the possibility of ensuring confidentiality through approaches, such as K-anonymity.

Table 1. Surveyed Papers

Ref.	Attacks					Defenses						General Challenges			
	Leak during transfer	**Leak during Central Preprocessing and Training**	**Training Data Extraction /Gradient Leakage**	**Data Poisoning**	**Model Extraction**	**FL**	**DP**	**Model Compression**	**Architecture**	**Homomorphic Encryption**	***K*-anonymity**	**Threat Intelligence**	**Computational Complexity**	**Explainable Results**	**Skewed and Biased Data**
[3]	-	-	×	√	×	×	×	-	√	×	×	√	×	√	√
[5]	-	-	×	×	×	×	-	-	√	-	-	-	√	-	√
[13]	×	×	×	√	×	√	-	-	√	-	-	-	×	-	×
[17]	×	×	×	×	-	√	-	-	√	-	-	-	×	-	√
[18]	×	√	√	×	×	√	√	√	√	√	×	-	-	-	-
[28]	-	-	-	-	-	-	-	√	√	-	-	-	√	-	-
[30]	-	-	√	-	-	×	×	×	√	×	×	-	-	-	-
[31]	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-
[32]	×	×	×	-	-	-	-	-	-	-	-	√	-	-	-
[33]	-	-	-	-	-	-	-	-	√	-	-	√	-	-	√
[38]	×	×	×	×	-	-	-	-	-	-	-	-	-	√	-
[40]	-	-	-	-	-	-	-	-	√	-	-	-	√	-	-
[41]	√	√	√	√	×	√	√	√	√	√	×	-	√	-	√
[42]	-	-	-	-	-	-	-	-	√	-	-	-	√	×	√
[45]	-	-	-	-	-	-	-	-	√	-	-	-	√	-	√
[47]	×	×	×	√	×	√	×	×	×	×	×	-	-	-	√
[48]	√	√	×	√	×	√	√	√	√	√	×	-	-	-	√
[49]	√	√	√	×	×	√	√	√	√	√	√	-	-	-	√
[64]	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
[65]	√	×	√	-	-	√	√	√	√	-	-	-	√	-	-
[67]	-	-	-	-	-	√	-	-	√	-	-	-	-	√	×
[69]	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
[84]	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
[85]	-	-	-	-	√	-	-	√	-	-	-	-	√	-	-
[88]	-	-	-	√	-	√	-	-	√	-	-	-	-	-	-
[93]	-	-	-	-	-	-	√	-	-	-	√	√	-	-	-
[97]	√	√	√	√	√	√	-	-	√	-	-	-	√	×	-
[103]	√	√	√	√	×	√	√	×	√	√	×	×	-	√	√
[104]	×	√	√	√	×	√	√	×	-	√	×	-	-	√	-
[106]	√	√	√	√	-	√	√	×	-	√	×	-	√	-	-
[107]	-	-	√	-	-	√	√	√	-	-	-	-	-	-	-

Zhang et al. [104] have provided a comprehensive overview of potential attacks, including poisoning and privacy leakage of FL. In a recent study, Nguyen et al. [65] presented an overview of distributed FL approaches, which were specifically designed for local devices with limited memory and computing capabilities. In [3], Apruzzese et al. address, in addition to a general overview and architecture, the threat of data poisoning and data sharing and confidentiality issues. Atefinia et al. [5] address the challenges of computational complexity and biased data by introducing a multi-architectural, modular DNN model. Du et al. compare in their work [18] the effects of different models, training, and pre- and post-processing to prevent gradient leakage in FL. Gorbett et al. focus on model compression, demonstrating how to reduce computational complexity while maintaining high accuracy [28]. Khraisat et al. [41] have conducted an in-depth examination of the diverse aggregation strategies, as well as a detailed analysis of the overarching taxonomy.

Lavaur et al. [13, 47, 48] elaborate the challenges associated with the heterogeneous distributed data, as well as on poisoning attacks and privacy breaches, within the context of the utilization of FL for IDS. Djaidja et al. [17] investigated the challenges posed by non-independently and non-identically distributed data in IDS applications and the performance of diverse FL techniques. Oki et al. [67] demonstrated that a distributed FL IDS at the edge of a network can achieve performance levels comparable to those of non-distributed systems. In their study, Zhang et al. [103] demonstrated that FIDSs could offer enhanced performance compared to classical IDSs in large-scale implementations.

Armed with the knowledge of existing works, we provide the various possible attacks on FL and evaluate their severity using failure models and effects analysis.

5 Threat Analysis for FL-Based IDS

In this section, first, we describe our threat assessment methodology and define our usage of important terms like "*threat*," "*misuse case*," "*safety*," and "*security*." Next, we consider threats that can harm the intrusion detection model. Finally, we consider threats that could leak private data.

5.1 Threat Assessment Methodology

We use the term *threat* according to the definition in the NIST Guide for Conducting Risk Assessments [37]. We quote this definition to make the paper self-contained.

> "Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, or modification of information, and/or denial of service."

A comprehensive identification and classification of potential risks for leakage of sensitive or PII was conducted, with the risk management standard ISO 31000 [35] serving as a general framework and the failure modes and effects analysis (FMEA and FMECA) standard IEC 60812 [29, 34] serving as a guideline. We use the systems engineering definitions of safety and security. *Safety* refers to how the system prevents harm to the environment, while *security* refers to how the system prevents harm from the environment. In this process, potential risks are analyzed based on their severity, occurrence, and detection. *Severity* refers to the effect of a risk ranging from no effect to exposing the client to loss, harm, or major disruption. *Occurrence* means the possibility of risk, ranging from unlikely to almost inevitable. *Detection* means the likelihood that a risk will be detected by process controls before the next process or before exposure to a client. The **Risk Priority Number (RPN)**, attributed to [6, 82], is a metric used to prioritize risks to indicate their criticality. The RPN of a specific risk [6, 15] is calculated as follows: RPN = Severity (S) × Occurrence (O) × Detection (D). The scales used for severity, occurrence, and detection are shown in Tables 2, 3 and 4, respectively.

Table 2. Severity Scale

Effect	Criteria: Severity of Effect	Ranking
Hazardous—Without Warning	May expose client to loss, harm, or major disruption—failure will occur without warning	10
Hazardous—With Warning	May expose client to loss, harm, or major disruption—failure will occur with warning	9
Very High	Major disruption of service involving client interaction, resulting in either associate re-work or inconvenience to client	8
High	Minor disruption of service involving client interaction and resulting in either associate re-work or inconvenience to clients	7
Moderate	Major disruption of service not involving client interaction and resulting in either associate re-work or inconvenience to clients	6
Low	Minor disruption of service not involving client interaction and resulting in either associate re-work or inconvenience to clients	5
Very Low	Minor disruption of service involving client interaction that does not result in either associate re-work or inconvenience to clients	4
Minor	Minor disruption of service not involving client interaction and does not result in either associate re-work or inconvenience to clients	3
Very Minor	No disruption of service noticed by the client in any capacity and does not result in either associate re-work or inconvenience to clients	2
None	No Effect	1

Table 3. Occurrence Scale

Possibility of Failure	Time Period	Per Item Failure Rates	Ranking
Very High: Failure is almost inevitable	More than once per day	≥1 in 2	10
	Once every 3–4 days	1 in 3	9
High: Generally associated with processes similar to previous processes that have often failed	Once every week	1 in 8	8
	Once every month	1 in 20	7
Moderate: Generally associated with processes similar to previous processes which have experienced occasional failures, but not in major proportions	Once every 3 months	1 in 80	6
	Once every 6 months	1 in 400	5
	Once a year	1 in 800	4
Low: Isolated failures associated with similar processes	Once every 1–3 years	1 in 1,500	3
Very Low: Only isolated failures associated with almost identical processes	Once every 3–6 years	1 in 3,000	2
Remote: Failure is unlikely No failures associated with almost identical processes	Once every 7+ years	1 in 6,000	1

Further, attacks that weaken the model's defensive capability and those that aim to extract the model have also been identified. The extraction of training data and disclosure of existing labels, as well as the number of data records used and their classification, must be considered sensitive.

The system structure, needs, requirements, and potential attacks were modeled using the UML-2-based **Systems Modeling Language (SysML)** [10]. The model encompassed the inter-dependencies between functional and technical requirements and the technical implementation. This approach facilitated a systematic and analytical documentation of the system [53] and enabled the simulation of the described threats, attacks, and defense mechanisms.

Table 4. Detection Scale

Detection	Criteria: Likelihood the existence of a defect will be detected by process controls before next or subsequent process, -OR- before exposure to a client	Ranking
Almost Impossible	No known controls available to detect failure mode	10
	Very remote likelihood current controls will detect failure mode	9
Remote	Remote likelihood current controls will detect failure mode	8
	Very low likelihood current controls will detect failure mode	7
Low	Low likelihood current controls will detect failure mode	6
	Moderate likelihood current controls will detect failure mode	5
	Moderately high likelihood current controls will detect failure mode	4
High	High likelihood current controls will detect failure mode	3
Very High	Very high likelihood current controls will detect failure mode	2
Almost Certain	Current controls almost certain to detect the failure mode Reliable detection controls are known with similar processes	1

Finally, we use the misuse case definition as stated by Sindre and Opdahl [77]. Here again, to make the article self-contained, we repeat their definitions verbatim.

Misuser. An actor that initiates misuse cases, either intentionally or inadvertently.

Misuse Case. A sequence of actions, including variants, that a system or other entity can perform, interacting with misusers of the entity and causing harm to some stakeholder if the sequence is allowed to complete.

Figure 2 gives an overview of examined misuse cases as a SysML diagram. In the SysML methodology, a Use Case diagram is defined as a context diagram, utilized for the purpose of modeling the overall relationships. In this particular instance, the system illustrated by the rectangle is representative of the entire IDS. The attacker is defined as an external entity to the system, and the ovals represent the defined misuse cases. More importantly, the model served as a communication and documentation tool within the team and for creating graphical representations. The threats could be categorized as follows: (a) threats that harm the model and (b) threats that may leak private data.

5.2 Threats Harming the Model: Data Leakage during Transfer

When analyzing IDS data centrally, whether by a service provider, cloud provider, or any other entity, sensitive data may be leaked during transfer. As a distributed IDS depends on external resources, such as the internet, for the transmission of data, this presents a dual concern in terms of safety and security. Figure 3 shows the dataflow and participating entities as an SysML activity diagram. An activity diagram is a visual representation of the activity of the underlying model, presented in the form of a swim lane diagram. In this particular instance, the diagram illustrates the information flow from a network operator utilizing a IDS via an intermediate network provider to a DNN Provider, with the possibility of an attacker intercepting the connection through a person-in-the-middle attack. The exposure of sensitive data to external parties is a potential safety issue, and the attack to

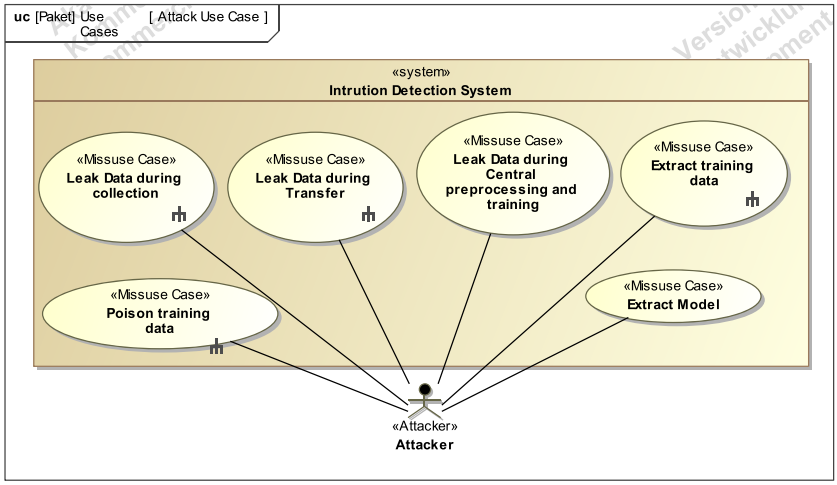

Fig. 2. SysML misuse cases diagram. SysML, Systems Modeling Language.

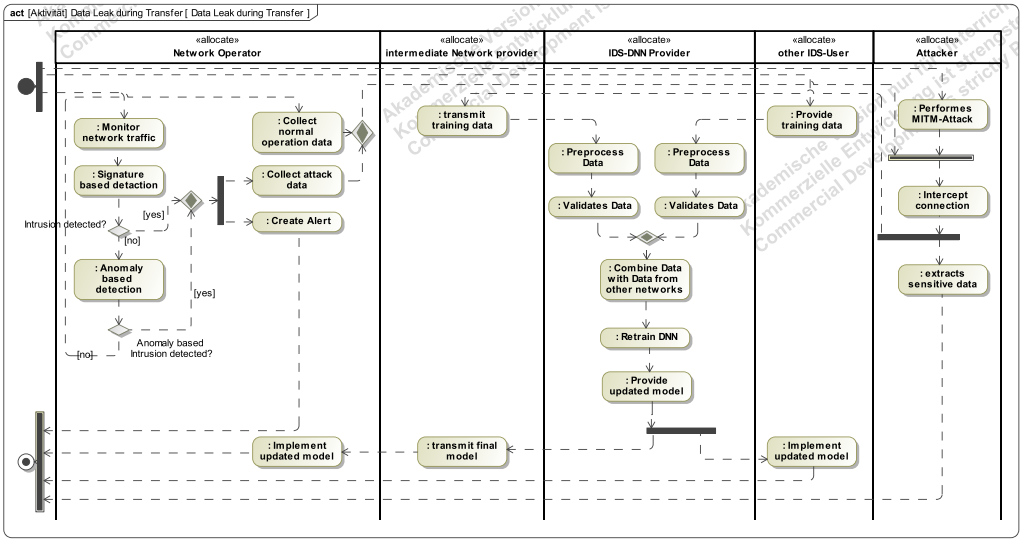

Fig. 3. SysML activity diagram Data Leak during transfer.

gain access to this data represents a security issue in the context of our system. Various network attacks, such as person-in-the-middle attacks, DNS poisoning, and BGP hijacking, could be used to gain access to the data.

The estimated severity of this attack is high $\frac{7}{10}$, as it would reveal sensitive information pertaining to a single organization. The probability of occurrence is estimated to be moderate $\frac{5}{10}$, as the attacker must be able to intercept the traffic at the appropriate time. This requires insights into the organization and advanced technical knowledge. The likelihood of detection of this attack is estimated to be low $\frac{6}{10}$, as there are some attack points external to our system.

5.3 Threats Leaking Private Data: Leakage during Central Preprocessing and Training

During analysis and central training, providers have access to sensitive raw data. This raises questions of trust and legality, particularly if confidential or private data is transferred to an external entity. It is often unclear who has access to the data, where it is processed, and where it is stored. Especially when subcontractors or hyperscalers' resources are used by the provider, it becomes difficult to identify and trace access and data storage or processing facilities. Note that, during every transfer the risks of Section 5.2 apply. The fact that a central entity processes confidential data for several organizations makes it an even more valuable target for attackers. Also, for various security reasons, governments and intelligence agencies have an interest in accessing sensitive data. Specifically, if processing takes place in a different country or jurisdiction than the resident of the data subject, local laws may allow governments to access the data, even without informing the data subject. For example, the United States Constitution guarantees privacy only for US citizens and not for foreigners [98]. Similar rules apply in other countries. Therefore, if data is transferred between different countries, intelligence agencies could analyze data from foreigners and share insights with allies to undermine restrictions related to their own citizens. Another possible attack vector is a compromised or adversarial server that interferes with the training process and attempts to extract all available information.

The estimated severity of this attack is very high $\frac{8}{10}$, as it would reveal sensitive information pertaining to all participating organizations. The probability of occurrence is estimated to be moderate $\frac{6}{10}$, as the service providers would suffer a loss of reputation and therefore have in place a control mechanism to protect unauthorized access. Nevertheless, the sensitive data of all organizations is freely available to all individuals with the requisite authorizations. The likelihood that current controls will detect this attack is estimated to be remote $\frac{8}{10}$, as this happens completely out of the control of the user organization.

6 Threats Targeting the FL and the Model

Finally, in this section, we will discuss the existing targeted data privacy threats against FL models and evaluate their severity for FIDS. We provide a proof-of-concept simulation attack on selected features of the CICIDS2017 dataset from the work by Sharafaldin et al. [74] to show the feasibility of such attacks. We note that, many of these attacks have not yet been described in literature for FIDS, but are quite likely to surface sooner or later.

6.1 Training Data Extraction

We will specifically focus on the problem of whether the network intrusion data will be leaked in such a way as to reveal sensitive information. Several recent studies have shown that large language models used for **Natural Language Processing (NLP)** [64] as well as **Computer Vision (CV)** models memorize a significant proportion of training data, including sensitive and personal information, which can be extracted from the trained model. Memorizing and extracting training data is generally easier with more complex (more parameters) and over-fitted models than with simpler or under-fitted ones. Current research [30] suggests that it is very difficult to prevent memorization and extraction without removing the unique/sensitive features from the training data itself. While there have been no successful attacks on FIDS models to date, such attacks cannot be ruled out.

Deep Gradient Leakage Attack is a specific technique [107] for training data extraction. In the context of distributed and FL, the exchange of gradients among participants is a critical component of the system. Research [100, 107] indicates that the properties of the training data, as well as complete samples of the training data, can be extracted from transmitted gradients. Recent studies have demonstrated that training data can also be extracted from graph-based FL via a method known as **Gradient Inversion Attack (GIA)** [78]. This particular type of GIA on FL can be mitigated by implementing post-processing gradients prior to transmission as discussed in [18].

Proof-of-Concept for IDS Training Data Extraction. We created a simulation based on the feature distributions from the public dataset [74]. The data samples were generated based on the actual log-normal distributions of

Table 5. Features Considered in Proof-of-Concept

Feature	Description
Flow Duration	Total duration of the flow in microseconds
Total Fwd/Bwd Pkts	Packet counts in forward and backward direction, respectively
Fwd/Bwd Pkt Len Mean	Mean of packet lengths in forward and backward direction, respectively
Flow Bytes/s	Rate of bytes in the flows or throughput
Flow Pkts/s	Packet rate
Flow IAT Mean	Measure timing gaps between consecutive packets
Fwd IAT Total	Gaps between forward packets only
Active Mean	Duration of active periods in flows

the selected features. Table 5 shows the descriptions of features used in our simulation. The traffic classes were labeled as "BENIGN," "DoS," "PortScan," "BruteForce," and "Infiltration." Five clients were considered where one of the client's gradients were intercepted by an attacker. The data at each client followed a heterogeneous non-IID class distribution.

We used a **Generative Adversarial Network (GAN)**-based network [51, 107] for reconstructing the data samples at the victim client, whose gradients were captured. The goal was to examine if any of the features could be faithfully recovered through this attack, which validates the attacks on FIDS. We used public repositories [105] provided by the authors and some amount of generative AI to assist in routine configuration setup of the neural models. The entire experiment was configured and controlled by us. The simulation ran for 25 trials and we reported the trials where some of the reconstructed features followed the same distribution as the original data.

FL Model. We used a multi-layer perceptron one input/output layer and 2 hidden layers with 64 neurons and ReLU activation function with a total of 5,189 learnable parameters. The number of training samples distributed across all the clients was 3,870 and the number of testing samples, which were not part of the training set, was 3,000.

GAN Model. To extract the features, we used a GAN model that uses a Generator network and a Discriminator network to reconstruct the training samples. The generator is a standard vanilla generator from [27] with one input layer, one hidden unit, and one output activation unit using the Sigmoid function. LeakyReLU is used in the input and hidden units. The discriminator has similar architecture with the hidden layer acting as a compressor. Both generator and discriminator use loss functions based on the Binary Cross Entropy (BCE loss), as per their respective functionality.

Experiments Conducted. We used mini-batch SGD for the optimization step and each local training run was on a batch of 32 samples for 100 epochs. The victim client was assumed to transmit the gradient updates in plain-text, which allowed the attacker to obtain the gradients. However, this was only done as a proof-of-concept and more sophisticated methods may also be used to compromise the client or capture the gradients. We report two successful attacks among all the runs conducted, where (a) three features, *Flow duration, Flow IAT Total, Active Mean*, were fully recovered by the attacker as shown in Figure 4 and (b) two features, *Fwd Pkt Len Mean, Flow IAT Mean*, were fully recovered as shown in Figure 5. The two figures display the normalized feature distribution values. A successful attack is indicated if there is significant overlap, i.e., more than 80% of the recovered features were extracted. In fact, a majority of the other features recovered by the attacker had similar distribution close to the real feature distribution, indicating that these attacks are practically feasible.

The estimated severity of this attack is moderate $\frac{6}{10}$, as it would reveal a portion of the sensitive information pertaining to all participating organizations. The probability of occurrence is estimated to be moderate $\frac{4}{10}$, given that this attack requires advanced knowledge and there is no guaranteed success of data extraction, nor could the attacker verify the authenticity of the extracted data without additional information. The likelihood of detection of this attack is estimated to be remote $\frac{8}{10}$, as this happens independent of the user organization and workflow.

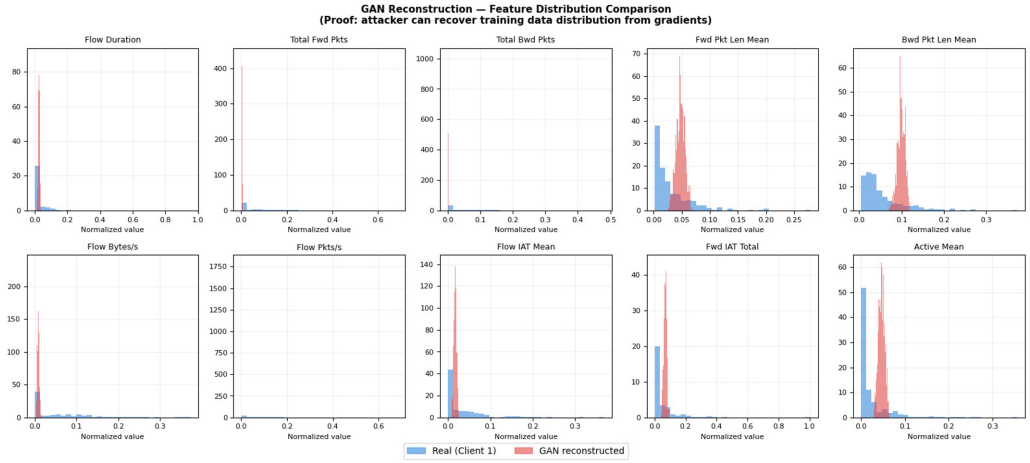

Fig. 4. Results for extraction of three features.

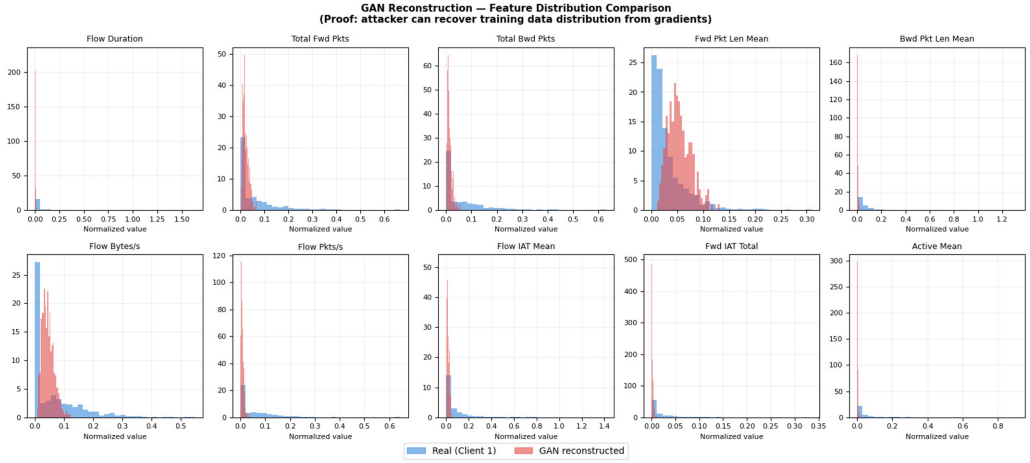

Fig. 5. Results for extraction of two features.

6.2 Data Poisoning

Data poisoning refers to the malicious manipulation of training data in order to weaken the model and enable later intrusion into target networks. This process involves intentionally introducing customized training data. For instance, this data may classify attacks as benign network traffic or weaken the model's detection rate in some other manner. Nguyen et al. [66], Vy et al. [92], and Thein et al. [87] demonstrated successful data poisoning attacks on FIDS. Figure 6 shows the SysML activity diagram, which depicts the interconnections and flow between the local network operator, network provider, IDS provider, other users, and the potential for an attacker to inject poisoned data into the learning pipeline. Later on, the model, which has been weakened in this way for certain attacks, can be deceived, allowing the attacker to penetrate the network undetected. Data poisoning can be used in combination with training data extraction to reveal the model structure or labels.

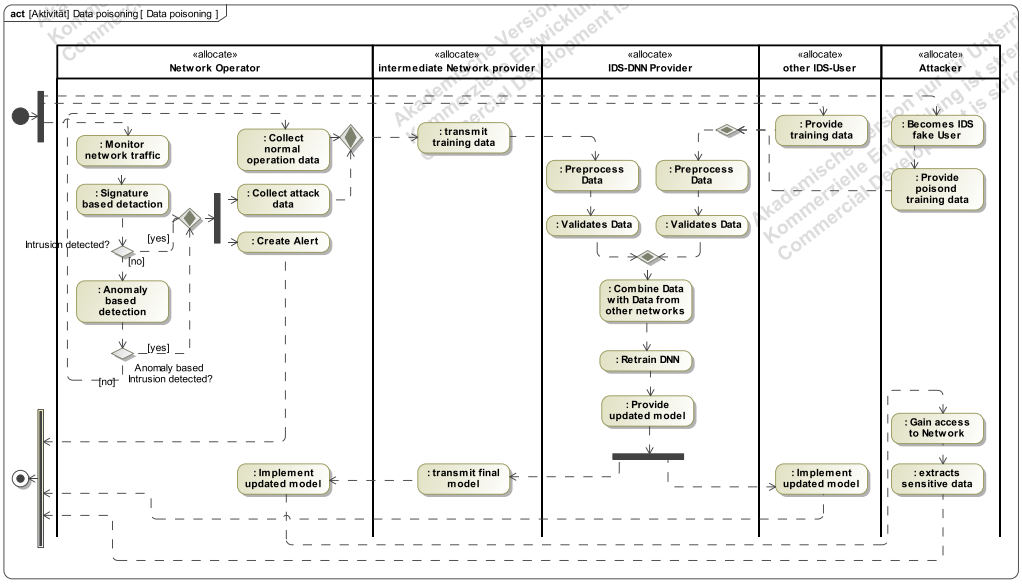

Fig. 6. SysML activity diagram data poisoning.

To address this challenge, a range of techniques have been developed to prevent or mitigate the risk of poisoning attacks. The application of these methods typically occurs during the data preprocessing stage, prior to aggregation. The aforementioned elements include, but are not limited to, distance-based analysis and averaging of the updates, as well as new graph-based clustering algorithms [71].

The estimated severity of this attack is high $\frac{7}{10}$, as it would weaken the model and enable attacks on all participating organizations. However, it will not reveal sensitive information itself. The probability of occurrence is estimated to be low (3/10), given that this attack requires advanced knowledge and a long time for preparation. Furthermore, no private data could be gained directly, and the success of a later attack cannot be guaranteed. The likelihood current controls will detect this attack is estimated to be very low $\frac{7}{10}$, as the raw training data is processed centrally and the responsibility for the delivered data lies with the participating organizations.

6.3 Model Extraction

This attack [85] is aimed at discovering or replicating the structure of the model, rather than extracting the training data itself. The detailed knowledge of the model can be employed to create a competing model with comparable performance without the original design and training effort. Furthermore, the knowledge can be utilized to identify and exploit weaknesses, which could be leveraged in subsequent attacks. It is also conceivable that model extraction could be employed as a technique to conceal the training data within the final model. Empirical evidence [31] indicates that extracted models can achieve greater accuracy than their original counterparts, while requiring significantly less computational complexity. This is particularly evident in complex ensemble model structures, which could be simplified and distilled to a single, more straightforward model.

The estimated severity of this attack is low $\frac{5}{10}$, as it would not reveal sensitive information but would enable the exploitation of additional attack vectors. The probability of occurrence is estimated to be low (3/10), given that this attack requires advanced knowledge and a long time for preparation. Furthermore, no private data could be gained directly, and the success of a later attack cannot be guaranteed. The likelihood of detection of this attack is estimated to be remote $\frac{8}{10}$, as this happens independent of the user organisation.

Table 6. Threat Priority

Threat	Severity (1–10)	Occurrence (1–10)	Detection (1–10)	RPN
Leak during Central Preprocessing and Training	8	6	8	384
Leak during Transfer	7	5	6	210
Training Data Extraction	6	4	8	192
Data Poisoning	7	3	7	147
Model Extraction	5	3	8	120

6.4 Threat Summary

The assessment and prioritization of threats in Table 6 was conducted based on the IEC 60812:2018 FMECA standard [34]. While the RPN represent a quantitative value, the value itself is not meaningful; rather, it serves to weight the various threats and to prioritize the actions and improvements to be taken.

7 Existing Defenses for FL

In this section, we provide an overview of the existing defenses for FL that can be considered, in the future, as defenses against data leakage in FIDS.

7.1 DP

DP [19] is a technique applied to the raw data during preprocessing and in other stages of the machine learning pipeline. In essence, a controlled quantity of noise is added to the data. The quantity of noise and, consequently, the degree of privacy is controlled by a privacy loss parameter ϵ. The introduction of noise into the data results in a reduction in both the utility and accuracy of the data. If DP is applied multiple times in the machine learning pipeline, the noise is accumulated, resulting in a corresponding reduction in accuracy. In FIDS, DP [49] may be applied during local data preprocessing, prior to the data being fed into the local learning pipeline or shared externally. DP could effectively preserve privacy in a controlled manner, but care must be taken to ensure that the utility is not adversely affected.

7.2 Homomorphic Encryption

Homomorphic encryption [106] is a method of training and computing on encrypted data without the need for decryption. This approach ensures the privacy of the data, but it is associated with low performance and poor computational efficiency. Consequently, the aforementioned solution is not currently applicable to FIDS.

7.3 Architectural Impact on Data Extraction

The model's architecture [100] can significantly impact data extraction. Research [18] has demonstrated that skip connections or net-in-net models can make the model more vulnerable to GIAs. Conversely, reducing bias, decreasing the size of convolutional kernels, or expanding padding can improve the model's resilience. In the case of an IDS, the number of attack-related network packets is significantly lower than the legitimate traffic on the network. Therefore, a classical comparison and optimization solely based on accuracy could be misleading.

7.4 Data Anonymization

Anonymization refers to the removal of identifiable information from the data, so as to break the association between the data and the data subject. Although data anonymization is intended to make re-identification impossible, it may still be possible to ascertain the identity of the data subject with the aid of additional data.

Table 7. FMECA

Threat	Sev. (1–10)	Occ. (1–10)	Det. (1–10)	RPN	Action	Sev. (1–10)	Occ. 1–10)	Det. (1–10)	RPN
Leak during Central Preprocessing and Training	8	6	8	384	FL, DP	5	3	8	120
Leak during Transfer	7	5	6	210	FL, DP	5	3	6	90
Training Data Extraction	6	4	8	192	DP, Compression	4	2	8	64
Data Poisoning	7	3	7	147	Verification, Randomization	7	2	4	56
Model Extraction	5	3	8	120	Model Compression	5	2	7	70

Pseudonymization [61] is the replacement of a privacy or confidential data object by a known placeholder. Note that, the organization responsible for pseudonymization is aware of the relationship between the placeholder and the original data subject. Consequently, the pseudonymization process could be reversed without information loss. Moreover, an attacker could potentially reveal the identity of the data subject or confidential data by analyzing patterns in the data or by utilizing additional data about the subject of interest. Therefore, data anonymization or pseudonymization are not adequate for the preservation of private or confidential data in the context of our use case.

7.5 K-anonymity

K-anonymity [2] is a technique that ensures that at least K datasets with the same attributes are present within the dataset. For an attacker with additional information about the data subject or the ability to query multiple times, K-anonymity is relatively weak, and may be less effective than DP. DP may be used as an additional layer of protection to still ensure privacy in the event that data would be extracted from the shared weights or from the trained model.

7.6 Unlearning

Data unlearning is a technique that could potentially be used after an accurate and consistent model has been attacked and compromised by data poisoning [95]. If the poisoned updates could be identified, the malicious data could be unlearned from the existing model. This avoids the computationally expensive process of completely retraining the model without the poisoned data. However, the federated unlearning process itself could also be poisoned again to prevent proper unlearning, which needs to be addressed by additional safeguards.

7.7 Summary of Defenses

In Table 7, we present a summary of the threats that have been studied and the proposed actions that have been formulated based on the prioritization that has been carried out according to the RPN. In accordance with the principles of FMECA, we have adopted a methodology for the analysis of cybersecurity threats.

Although the RPN cannot be interpreted as an absolute measure, a clear trend can be identified from the actions investigated and proposed. It appears that the priority goal of significantly reducing the risk of data leakage during central preprocessing and training as well as during transfer can be achieved using the techniques investigated. It is noteworthy that the less urgent threats can also be addressed through these methodologies. In summary, the most promising privacy-preserving techniques would first be the use of FL to not share raw data at all. This can be combined with DP [50] as a preprocessing technique to mitigate the impact of leaks and quantization [39, 96, 101] as a post-processing technique to make the system resilient against data leakages. The

investigated techniques will be applied in a thorough way during the deep learning pipeline and together with appropriate training strategies such as local training for multiple epochs to ensure privacy and efficiency.

8 Conclusion and Future Work

In this work, we addressed the problem of data privacy in cyber-threat intelligence sharing with a focus on IDS. The surest way to prevent the leakage of private or confidential data is to keep the raw data local and avoid sharing it with other parties. This can be achieved by using a FL approach, where the model is partially trained locally and only the model weights are transferred and shared. Weights in FL nodes can be exchanged directly between peers in decentralized FL or coordinated and controlled by a server in centralized FL.

FL does not share training data with the central server or other clients, but a hostile server can still perform a GIA to extract the training data through shared gradients. Research has demonstrated that the effectiveness of the extraction process is influenced by various factors, including the model's structure, the type of training utilized, the preprocessing of the training data, and the post-processing of the gradients. Studies [18] have also shown that performing local training in a multiple steps (batches) and across several epochs can make training data extraction more difficult.

Only a few studies [56] discuss the use of FL for IDS. Attacks on FL, such as, GIAs, have been shown on image, CV and NLP problems, but are yet to be demonstrated on IDS. We are not yet aware of any open-world study or implementation that covers the discussed aspects of privacy and security of FL on IDS, as well as the aspects necessary for practical implementation, such as data freshness and handling by network operators.

A lot of work remains to be done. One work is to investigate what sensitive information about an organization infrastructure gets disclosed when network features are revealed to external entities. Another direction of research is understanding whether new forms of data leakage are possible when FL is used for intrusion detection. It also remains to be seen whether the addition of privacy preserving technologies such as DP [19] for FL provides stronger privacy guarantees. We need extensive experimental evaluation to investigate the various attack and mitigation strategies in IDS.

Another research direction involves exploring methods for efficient adaptation of these models generated through FL to new attack data that shows significant drift or exhibits unseen feature variations. Addressing such data drifts is non-trivial. When considered in a complex distributed environment, this problem opens up new avenues for both researchers and practitioners.

References

[1] M. Alloghani, M. M. Alani, D. Al-Jumeily, T. Baker, J. Mustafina, A. Hussain, and A. J. Aljaaf. 2019. A systematic review on the status and progress of homomorphic encryption technologies. *Journal of Information Security and Applications* 48 (2019), 102362.

[2] R. Anderson. 2020. *Security Engineering: A Guide to Building Dependable Distributed Systems* (3rd ed.). Wiley.

[3] G. Apruzzese, P. Laskov, E. Montes De Oca, W. Mallouli, L. Brdalo Rapa, A. V. Grammatopoulos, and F. Di Franco. 2023. The role of machine learning in cybersecurity. *Digital Threats: Research and Practice* 4, 1 (Mar. 2023), 1–38.

[4] G. Apruzzese, L. Pajola, and M. Conti. 2022. The cross-evaluation of machine learning-based network intrusion detection systems. *IEEE Transactions on Network and Service Management* 19, 4 (2022), 5152–5169.

[5] R. Atefinia and M. Ahmadi. 2021. Network intrusion detection using multi-architectural modular deep neural network. *The Journal of Supercomputing* 77, 4 (Apr. 2021), 3571–3593.

[6] AIAG (Automotive Industry Action Group). 1993. *Potential Failure Mode and Effects Analysis (FMEA) Reference Manual* (1st ed.).

[7] K. Baker. 2023. What Is Cyber Threat Intelligence? Retrieved October 23, 2024 from https://www.crowdstrike.com/en-us/cybersecurity-101/threat-intelligence/

[8] R. J. Bayardo and R. Agrawal. 2005. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering (ICDE '05)* IEEE, 217–228.

[9] K. Bonawitz, P. Kairouz, B. Mcmahan, and D. Ramage. 2022. Federated learning and privacy. *Communications of the ACM* 65, 4 (Mar. 2022), 90–97.

[10] J. M. Borky and T. H. Bradley. 2019. *Effective Model-Based Systems Engineering*. Springer.

[11] Broadcom. 2023. Carbon Black. Retrieved May 21, 2025 from https://www.broadcom.com/products/carbon-black

[12] A. L. Buczak and E. Guven. 2015. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials* 18, 2 (2015), 1153–1176.
[13] Y. Busnel and L. Lavaur. 2024. Tutorial: Federated learning × security for network monitoring. In *Proceedings of the 2024 IEEE 44th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 13–16.
[14] Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan. 2020. Intrusion detection for wireless edge networks based on federated learning. *IEEE Access* 8 (2020), 217463–217472.
[15] D. Croft-Bednarski. 2024. *FMEA Template—(Free Excel & Google Sheet + Example)*. Section: Risk Management. Retrieved from https://www.learnleansigma.com/template/fmea-template/
[16] L. Dandurand, A. Kaplan, P. Kácha, Y. Kadobayashi, A. Kompanek, T. Lima, T. Millar, J. Nazario, R. Perlotto, and W. Young. 2014. *Standards and Tools for Exchange and Processing of Actionable Information. European Union Agency for Network and Information Security (ENISA)*, Tech. Rep.
[17] T. E. T. Djaidja, B. Brik, A. Boualouache, S. M. Senouci, and Y. Ghamri-Doudane. 2024. Federated learning for 5G and beyond, a blessing and a curse—An experimental study on intrusion detection systems. *Computers & Security* 139 (Apr. 2024), 103707.
[18] J. Du, J. Hu, Z. Wang, P. Sun, N. Z. Gong, and K. Ren. 2024. SoK: Gradient Leakage in Federated Learning. arXiv:2404.05403. Retrieved from https://arxiv.org/abs/2404.05403
[19] C. Dwork. 2006. Differential privacy. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*. Springer, 1–12.
[20] Y. Fan, Y. Li, M. Zhan, H. Cui, and Y. Zhang. 2020. IoTDefender: A federated transfer learning intrusion detection framework for 5G IoT. In *Proceedings of the 2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*. IEEE, 88–95.
[21] M. Fang, X. Cao, J. Jia, and N. Gong. 2020. Local model poisoning attacks to Byzantine-Robust federated learning. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security '20)*, 1605–1622.
[22] For Cybersecurity (ENISA), TEUA. 2024. Cyber threats. Retrieved October 26, 2024 from https://www.enisa.europa.eu/topics/cyber-threats
[23] J. H. Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38, 4 (2002), 367–378.
[24] I. M. Fund. 2024. *Global Financial Stability Report, April 2024: The Last Mile: Financial Vulnerabilities and Risks*. International Monetary Fund.
[25] S. Garcia, M. Grill, J. Stiborek, and A. Zunino. 2014. An empirical comparison of botnet detection methods. *Computers & Security* 45 (2014), 100–123.
[26] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security* 28, 1–2 (2009), 18–28.
[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2020. Generative adversarial networks. *Communications of the ACM* 63, 11 (2020), 139–144.
[28] M. Gorbett. 2024. *Towards Fair and Efficient Distributed Intelligence*. Ph.D. thesis, Colorado State University.
[29] T. Gueorguiev, M. Kokalarov, and B. Sakakushev. 2020. Recent trends in FMEA methodology. In *Proceedings of the 2020 7th International Conference on Energy Efficiency and Agricultural Engineering (EE&AE)*. IEEE, 1–4.
[30] J. Hartley, P. P. Sanchez, F. Haider, and S. A. Tsaftaris. 2023. Neural networks memorise personal information from one sample. *Scientific Reports* 13, 1 (Dec. 2023), 21366.
[31] G. Hinton, O. Vinyals, and J. Dean. 2015. Distilling the knowledge in a neural network. arXiv:1503.02531. Retrieved from https://arxiv.org/abs/1503.02531
[32] M. Horák, V. Stupka, and M. Husák. 2019. GDPR compliance in cybersecurity software: A case study of DPIA in information sharing platform. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*. ACM, 1–8.
[33] M. Husák, P. Sokol, M. Žádník, V. Bartoš, and M. Horák. 2023. Lessons learned from automated sharing of intrusion detection alerts: The case of the SABU platform. *Digital Threats: Research and Practice* 4, 4 (Dec. 2023), 1–11.
[34] IEC (International Electrotechnical Commission). 2018. Failure Modes and Effects Analysis (FMEA and FMECA). Retrieved from https://webstore.iec.ch/en/publication/26359
[35] ISO (International Organization for Standardization). 2018. Risk Management. Retrieved from https://www.iso.org/standard/65694.html
[36] C. S. Johnson, M. L. Badger, D. A. Waltermire, J. Snyder, and C. Skorupka. 2016. *Guide to Cyber Threat Information Sharing*. DOI: https://doi.org/10.6028/NIST.SP.800-150
[37] Joint Task Force Transformation Initiative. 2012. *Guide for Conducting Risk Assessments*. Tech. Rep. NIST SP 800-30r1. National Institute of Standards and Technology.
[38] V. Jüttner, M. Grimmer, and E. Buchmann. 2023. ChatIDS: Explainable cybersecurity using generative AI. arXiv:2306.14504. Retrieved from https://arxiv.org/abs/2306.14504
[39] T. Kang, L. Liu, H. He, J. Zhang, S. Song, and K. B. Letaief. 2024. The effect of quantization in federated learning: A Rényi differential privacy perspective. In *Proceedings of the 2024 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. IEEE, 233–238.

[40] A. M. Karimi, Q. Niyaz, Sun Weiqing, A. Y. Javaid, and V. K. Devabhaktuni. 2016. Distributed network traffic feature extraction for a real-time IDS. In *Proceedings of the 2016 IEEE International Conference on Electro Information Technology (EIT)*. IEEE, 0522–0526.

[41] A. Khraisat, A. Alazab, S. Singh, T. Jan, Jr., and A. Gomez. 2025. Survey on federated learning for intrusion detection system: Concept, architectures, aggregation strategies, challenges, and future directions. *ACM Computing Surveys* 57, 1 (Jan. 2025), 1–38.

[42] K. Kim, M. E. Aminanto, and H. C. Tanuwidjaja.2018. Network intrusion detection using deep learning: A feature learning approach. In *Springer Briefs on Cyber Security Systems and Networks*, Springer.

[43] S. Kim, H. Cai, C. Hua, P. Gu, W. Xu, and J. Park. 2020. Collaborative anomaly detection for internet of things based on federated learning. In *Proceedings of the 2020 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 623–628.

[44] D. Kingma and P. B. J. Adam. 2015. A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR '15)*.

[45] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto. 2021. Attack classification of an intrusion detection system using deep learning and hyperparameter optimization. *Journal of Information Security and Applications* 58 (May 2021), 102804.

[46] L. Lamport, R. Shostak, and M. Pease. 1982. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems* 4, 3 (1982), 203–226.

[47] L. Lavaur, Y. Busnel, and F. Autrel. 2024. Demo: Highlighting the limits of federated learning in intrusion detection. In *Proceedings of the 2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1416–1419.

[48] L. Lavaur, M.-O. Pahl, Y. Busnel, and F. Autrel. 2022. The evolution of federated learning-based intrusion detection and mitigation: A survey. *IEEE Transactions on Network and Service Management* 19, 3 (Sep. 2022), 2309–2332.

[49] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (May 2020), 50–60.

[50] Z. Li, H. Zhao, B. Li, and Y. Chi. 2022. SoteriaFL: A unified framework for private federated learning with communication compression. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22)*, Curran Associates.

[51] J. Q. Lim and C. S. Chan. 2021. From gradient leakage to adversarial attacks in federated learning. In *2021 IEEE International Conference on Image Processing (ICIP)*.

[52] Y. Lindell. 2020. Secure multiparty computation. *Communications of the ACM* 64, 1 (2020), 86–96.

[53] A. M. Madni, N. Augustine, and M. Sievers (Eds.). 2023. *Handbook of Model-Based Systems Engineering*. Springer.

[54] D. Malkhi and M. Reiter. 1997. Unreliable intrusion detection in distributed computations. In *Proceedings 10th Computer Security Foundations Workshop* IEEE, 116–124.

[55] D. Malkhi and M. Reiter. 1998. Byzantine quorum systems. *Distributed Computing* 11, 4 (1998), 203–213.

[56] Q. Mao, X. Lin, W. Xu, Y. Qi, X. Su, G. Li, and J. Li. 2025. FeCograph: Label-aware federated graph contrastive learning for few-shot network intrusion detection. *IEEE Transactions on Information Forensics and Security* 20 (2025), 2266–2280.

[57] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean. 2000. Boosting algorithms as gradient descent. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS '13)*, 512–518.

[58] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS '17)*, Vol. 54. PMLR, 1273–1282.

[59] Microsoft. 2024. Microsoft Digital Defense Report 2024. Retrieved from https://www.microsoft.com/en-us/security/security-insider/threat-landscape/microsoft-digital-defense-report-2024

[60] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai. 2018. Kitsune: An ensemble of autoencoders for online network intrusion detection. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS '18)*. The Internet Society.

[61] M. Monteiro, F. F. Correia, and P. G. G. Queiroz. 2024. Patterns for anonymization, pseudonymization and perturbation: Focus group report. In *Proceedings of the 29th European Conference on Pattern Languages of Programs, People, and Practices*. ACM, 1–4.

[62] V. Mugunthan, A. Polychroniadou, D. Byrd, and T. H. Balch. 2019. SMPAI: Secure multi-party computation for federated learning. In *Proceedings of the NeurIPS 2019 Workshop on Robust AI in Financial Services*, Vol. 21. MIT Press.

[63] S. Mukkamala, A. H. Sung, and A. Abraham. 2005. Intrusion detection using an ensemble of intelligent paradigms. *Journal of Network and Computer Applications* 28, 2 (2005), 167–182.

[64] M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee. 2023. Scalable extraction of training data from (production) language models. arXiv:2311.17035. Retrieved from https://arxiv.org/abs/2311.17035

[65] Q. Nguyen, H. H. Pham, K.-S. Wong, P. Le Nguyen, T. T. Nguyen, and M. N. Do. 2024. FedDCT: Federated learning of large convolutional neural networks on resource-constrained devices using divide and collaborative training. *IEEE Transactions on Network and Service Management* 21, 1 (Feb. 2024), 418–436.

[66] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi. 2020. Poisoning attacks on federated learning-based IoT intrusion detection system. In *Proceedings of the 2020 Workshop on Decentralized IoT Systems and Security*. Internet Society.

[67] A. Oki, Y. Ogawa, K. Ota, and M. Dong. 2024. Evaluation of applying federated learning to distributed intrusion detection systems through explainable AI. *IEEE Networking Letters* 6, 3 (Sep. 2024), 198–202.

[68] L. T. Phong and T. T. Phuong. 2019. Privacy-preserving deep learning via weight transmission. *IEEE Transactions on Information Forensics and Security* 14, 11 (2019), 3003–3015.
[69] J. V. Rani, H. A. Saeed Ali, and A. Jakka. 2023. IoT network intrusion detection: An explainable AI approach in cybersecurity. In *Proceedings of the 2023 4th International Conference on Communication, Computing and Industry 6.0 (C216)*. IEEE, 1–6.
[70] S. J. Reddi, S. Kale, and S. Kumar. 2019. On the convergence of Adam and beyond. arXiv:1904.09237. Retrieved from https://arxiv.org/abs/1904.09237
[71] H. Rezaei, R. Taheri, M. Shojafar, and C. H. Foh. 2025. GRAF-IDS: Graph-based clustering as aggregation for federated intrusion detection system in IoT network. *Neural Computing and Applications* 37 (Jun. 2025), 18401–18423.
[72] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho. 2019. A survey of network-based intrusion detection data sets. *Computers & Security* 86 (2019), 147–167.
[73] H. Robbins and S. Monro. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics* 22 (1951), 400–407.
[74] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*. INSTICC, SciTePress, 108–116.
[75] R. Shokri and V. Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 909–910.
[76] T. Shon and J. Moon. 2007. A hybrid machine learning approach to network anomaly detection. *Information Sciences* 177, 18 (2007), 3799–3821.
[77] G. Sindre and A. L. Opdahl. 2005. Eliciting security requirements with misuse cases. *Requirements Engineering* 10, 1 (Jan. 2005), 34–44.
[78] D. A. Sinha, R. Du, Y. Liu, A. Markopolou, and Y. Shen. 2025. Gradient inversion attack on graph neural networks. arXiv:2411.19440. Retrieved from https://arxiv.org/abs/2411.19440
[79] J. So, B. Güler, and A. S. Avestimehr. 2020. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications* 39, 7 (2020), 2168–2181.
[80] R. Sommer and V. Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, 305–316.
[81] P. Spadaccino and F. Cuomo. 2022. Intrusion detection systems for IoT: Opportunities and challenges offered by edge computing. *ITU Journal on Future and Evolving Technologies* 3, 2 (Sep. 2022), 408–420.
[82] D. H. Stamatis. 1995. *Failure Mode and Effect Analysis: FMEA from Theory to Execution*. ASQC Quality Press.
[83] J. Stock, O. Hauke, J. Weißmann, and H. Federrath. 2023. The applicability of federated learning to official statistics. In *Intelligent Data Engineering and Automated Learning (IDEAL '23)*. P. Quaresma, D. Camacho, H. Yin, T. Gonçalves, V. Julian, and A. J. Tallón-Ballesteros (Eds.), Lecture Notes in Computer Science, Vol. 14404, Springer Nature, pp. 70–81.
[84] M. Sundararajan, A. Taly, and Q. Yan. 2017. Axiomatic attribution for deep networks. arXiv:1703.01365. Retrieved from https://arxiv.org/abs/1703.01365
[85] T. Takemura, N. Yanai, and T. Fujiwara. 2020. Model extraction attacks on recurrent neural networks. *Journal of Information Processing* 28 (2020), 1010–1024.
[86] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*. IEEE, 1–6.
[87] T. T. Thein, Y. Shiraishi, and M. Morii. 2024. Personalized federated learning-based intrusion detection system: Poisoning attack and defense. *Future Generation Computer Systems* 153 (Apr. 2024), 182–192.
[88] Z. Tian, L. Cui, J. Liang, and S. Yu. 2023. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *ACM Computing Surveys* 55, 8 (Aug. 2023), 1–35.
[89] P. T. Tran and L. T. Phong. 2019. On the convergence proof of AMSGrad and a new version. *IEEE Access* 7 (2019), 61706–61716.
[90] University of California, Irvine. 1999. KDD'99 dataset. Retrieved April 22, 2022 from https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[91] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer. 2015. Taxonomy and survey of collaborative intrusion detection. *ACM Computing Surveys* 47, 4 (2015), 1–33.
[92] N. C. Vy, N. H. Quyen, P. T. Duy, and V.-H. Pham. 2021. Federated learning-based intrusion detection in the context of IIoT networks: Poisoning attack and defense. In *Network and System Security*. M. Yang, C. Chen, and Y. Liu (Eds.), Lecture Notes in Computer Science, Vol. 13041, Springer, pp. 131–147.
[93] T. D. Wagner, K. Mahbub, E. Palomar, and A. E. Abdallah. 2019. Cyber threat intelligence sharing: Survey and research directions. *Computers & Security* 87 (Nov. 2019), 101589.
[94] K. Wang and S. J. Stolfo. 2004. Anomalous payload-based network intrusion detection. In *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*. Springer, 203–222.
[95] W. Wang, Q. Ma, Z. Zhang, Y. Liu, Z. Liu, and M. Fang. 2025. Poisoning attacks and defenses to federated unlearning. arXiv:2501.17396. Retrieved from https://arxiv.org/abs/2501.17396

[96] Z. Wang, Q. Kang, X. Zhang, and Q. Hu. 2022. Defense strategies toward model poisoning attacks in federated learning: A survey. In *Proceedings of the 2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 548–553.
[97] A. A. Wardana and P. Sukarno. 2024. Taxonomy and survey of collaborative intrusion detection system using federated learning. *ACM Computing Surveys* 57, 4 (Dec. 2024), 1–36.
[98] E. Watt. 2017. The right to privacy and the future of mass surveillance. *The International Journal of Human Rights* 21, 7 (Sep. 2017), 773–799.
[99] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.
[100] R. Xu, N. Baracaldo, and J. Joshi. 2021. Privacy-preserving machine learning: Methods, challenges and directions. arXiv:2108.04417. Retrieved from https://arxiv.org/abs/2108.04417
[101] G. Yan, T. Li, K. Wu, and L. Song. 2024. Killing two birds with one stone: Quantization achieves privacy in distributed learning. *Digital Signal Processing* 146 (2024), 104353.
[102] A. C. Yao. 1982. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS '82)*. IEEE, 160–164.
[103] H. Zhang, J. Ye, W. Huang, X. Liu, and J. Gu. 2025. Survey of federated learning in intrusion detection. *Journal of Parallel and Distributed Computing* 195 (Jan. 2025), 104976.
[104] Y. Zhang, D. Zeng, J. Luo, X. Fu, G. Chen, Z. Xu, and I. A. King. 2024. Survey of trustworthy federated learning: Issues, solutions, and challenges. *ACM Transactions on Intelligent Systems and Technology* 15, 6 (Dec. 2024), 1–47.
[105] B. Zhao, K. R. Mopuri, and H. Bilen. 2020. Improved-Deep-Leakage-from-Gradients. Retrieved from https://github.com/PatrickZH/Improved-Deep-Leakage-from-Gradients
[106] J. Zhou, S. Chen, Y. Wu, H. Li, B. Zhang, L. Zhou, Y. Hu, Z. Xiang, Z. Li, N. Chen, et al. 2024. PPML-Omics: A privacy-preserving federated machine learning method protects patients' privacy in omic data. *Science Advances* 10 (2024).
[107] L. Zhu, Z. Liu, and S. Han. 2019. Deep leakage from gradients. arXiv:1906.08935. Retrieved from https://arxiv.org/abs/1906.08935

A Appendix

Table A1. List of Abbreviations

Abbreviation	**Definition**
BCE	Binary Cross Entropy
BGP	Border Gateway Protocol
CIDS	Collaborative Intrusion Detection System
CTI	Cyber Threat Intelligence
CUI	Controlled Unclassified Information
CV	Computer Vision
DNS	Domain Name System
DoS	Denial of Service
DP	Differential Privacy
ENISA	European Network and Information Security Agency
FIDS	Federated Learning-Based IDS
FL	Federated Learning
FMEA	Failure Mode and Effects Analysis
FMECA	Failure Modes Effects and Criticality Analysis
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
GIA	Gradient Inversion Attack
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IPS	Intrusion Prevention System
ISO	International Organization for Standardization
MLP	Multi-Layer Perceptron
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
PII	Personally Identifiable Information
RPN	Risk Priority Number
SGD	Stochastic Gradient Descent
SMC	Secure Multi-Party Computation
SoK	Systematization-of-Knowledge
SysML	Systems Modeling Language
UML	Unified Modeling Language

Received 31 August 2025; revised 3 April 2026; accepted 12 June 2026