# FedNIDS: A Federated Learning Framework for Packet-Based Network Intrusion Detection System

QUOC H. NGUYEN and SOUMYADEEP HORE, University of South Florida, Tampa, Florida, USA
ANKIT SHAH, Indiana University, Bloomington, Indiana, USA
TRUNG LE, University of South Florida, Tampa, Florida, USA
NATHANIEL D. BASTIAN, United States Military Academy, West Point, New York, USA

Network intrusion detection systems (NIDS) play a critical role in discerning between benign and malicious network traffic. Deep neural networks (DNNs), anchored on large and diverse datasets, exhibit promise in enhancing the detection accuracy of NIDS by capturing intricate network traffic patterns. However, safeguarding distributed computer networks against emerging cyber threats is increasingly challenging. Despite the abundance of diverse network data, decentralization persists due to data privacy and security concerns. This confers an asymmetric advantage to adversaries, as distributed networks face the formidable task of securely and efficiently sharing non-independently and identically distributed data to counter cyber-attacks. To address this, we propose the federated NIDS (*FedNIDS*), a novel two-stage framework that combines the power of federated learning and DNNs. It aims to enhance the detection accuracy of known attacks, robustness and resilience to novel attack patterns, and privacy preservation, using packet-level granular data. In the first stage, a global DNN model is collaboratively trained on distributed data, and the second stage adapts it to novel attack patterns. Our experiments on real-world intrusion datasets demonstrate the effectiveness of *FedNIDS* by achieving an average F1 score of 0.97 across distributed networks and quickly disseminating novel attack information within four rounds of communication.

CCS Concepts: • **Security and privacy** → **Network security**; **Intrusion/anomaly detection and malware mitigation**; • **Computing methodologies** → **Machine learning algorithms**;

Additional Key Words and Phrases: Federated learning, network intrusion detection systems, distributed and private network data, packet-based NIDS, novel attack detection

**ACM Reference format:**
Quoc H. Nguyen, Soumyadeep Hore, Ankit Shah, Trung Le, and Nathaniel D. Bastian. 2025. FedNIDS: A Federated Learning Framework for Packet-Based Network Intrusion Detection System. *Digit. Threat. Res. Pract.* 6, 1, Article 4 (February 2025), 23 pages.
https://doi.org/10.1145/3696012

Quoc H. Nguyen and Soumyadeep Hore contributed equally to this research.
The views and conclusions expressed in this article are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, U.S. Army, U.S. Department of Defense, or U.S. Government.
This work was supported in part by the U.S. Military Academy (USMA) under Cooperative Agreement No. W911NF-22-2-0045, as well as the U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory under Support Agreement No. USMA 21050.
Authors' Contact Information: Quoc H. Nguyen, University of South Florida, Tampa, Florida, USA; e-mail: nguyenq29@usf.edu; Soumyadeep Hore, University of South Florida, Tampa, Florida, USA; e-mail: soumyadeep@usf.edu; Ankit Shah (corresponding author), Indiana University, Bloomington, Indiana, USA; e-mail: ankit@iu.edu; Trung Le, University of South Florida, Tampa, Florida, USA; e-mail: tqle@usf.edu; Nathaniel D. Bastian, United States Military Academy, West Point, New York, USA; e-mail: nathaniel.bastian@westpoint.edu.


ACM 2576-5337/2025/2-ART4
https://doi.org/10.1145/3696012

## 1 Introduction

The proliferation of digital systems and the increased reliance on interconnected networks have elevated the significance of ensuring the security of computer systems and data. **Network intrusion detection systems (NIDS)** play a pivotal role in identifying potential threats and attacks within computer networks. These systems employ signature-based [16, 30] and anomaly-based [17, 51] techniques to detect attacks. With advancements in algorithmic knowledge and the availability of high-performance computing resources, **deep learning (DL)** methods have taken center stage in the development of state-of-the-art NIDS. DL-based NIDS are developed using both supervised [4, 8] and unsupervised [24, 52] training methods. With the assistance of labeled training data containing attack categories, the supervised learning paradigm has demonstrated superior performance in identifying previously known attacks when compared to the unsupervised learning approach. The type of data used for training DL-based NIDS can be broadly categorized into flow-based and packet-based data. While flow-based analysis provides valuable insights into network activity and has seen advancements through the incorporation of application metadata, our study focuses on leveraging the rich, granular information available in packet data. Packet-based analysis allows for more precise and timely detection of security incidents, complementing the high-level overview offered by flow data [11, 18, 20].

DL-based NIDS rely on the availability of large, consolidated datasets to learn the underlying patterns of benign and malicious network traffic. However, accumulating and consolidating data across different entities in distributed computer networks pose challenges in various situations: (i) Different branches within an organization, such as those in banks, hospitals, or the Department of Defense, often handle sensitive and confidential information that cannot be shared in a central location to train DL models. (ii) Many regions, like the European Union, and industries, such as healthcare under HIPAA regulations, enforce strict data privacy and security regulations that restrict the movement and sharing of data. (iii) Networks can exhibit significant variations in terms of their size and traffic patterns, and aggregating their data to train a centralized model may not necessarily result in superior detection accuracy across different network segments. (iv) Transmitting large volumes of data to a centralized server for DL model training may demand substantial bandwidth and introduce latency. Hence, this approach may not be well-suited for environments constrained by limited bandwidth and low-latency requirements, such as communication in military operations or real-time monitoring and transmission using **Internet of Things (IoT)** in an industrial setting. Another challenge in NIDS is the delayed identification of new attack patterns, such as zero-day and evasion attacks. This gap becomes particularly significant in a distributed setting where various entities cannot share novel attack data among themselves. This situation necessitates an adaptive mechanism for rapidly disseminating the knowledge from the affected participants. These challenges underline the need for alternative approaches in DL-based NIDS, which can address data privacy concerns, including regulatory constraints, and the variability in types of attacks experienced by the distributed entities while optimizing network intrusion detection.

The federated learning paradigm provides a compelling solution to the aforementioned challenges, empowering the collaborative training of models across multiple clients (distributed networks) without requiring centralization of data. This collaborative paradigm preserves data privacy and security and involves clients training their local models on their private data. Subsequently, the clients share the model updates in a secure and aggregated manner [13]. We introduce a novel framework called the **federated network intrusion detection system (FedNIDS)**, which leverages the synergy of federated learning and **deep neural networks (DNNs)** to enhance the accuracy and robustness of the NIDS against an evolving adversarial environment. *FedNIDS* aims to address the challenges in accurate detection of malicious network activities while upholding data privacy and security by training the DNNs on decentralized and privacy-sensitive packet-based data sources.

The primary contribution of this study is the development of the federated learning framework, *FedNIDS*, designed to operate in a distributed environment, utilizing granular packet-level data. The objective of this framework is to identify known attack patterns and adapt to detect emerging threats. It consists of two essential

stages: federated DNN pre-training and federated novel attack fine-tuning. In the first stage, the framework trains a global DNN-based binary classifier by leveraging decentralized data from participating clients. The second stage involves fine-tuning the pre-trained global model from the first stage with novel attack samples, as observed by the client(s). Our study is the first to address the issue of **non-independently and identically distributed (non-IID)** data across different clients for packet-based NIDS. We employ publicly available benchmark network intrusion detection datasets for experimentation and evaluation. The results illustrate the robustness and resilience of our *FedNIDS* framework in a distributed environment when encountering various forms of attacks, including known and unknown threats, such as zero-day and adversarially perturbed evasion attacks. These findings collectively contribute to the advancement of NIDS technology, enhancing their ability to identify and mitigate a broad spectrum of network attacks, particularly in situations where network packet data among different clients is non-IID.

The structure of the remainder of the article is as follows: Section 2 reviews the existing literature on NIDS and federated learning. Section 3 outlines our methodological framework, *FedNIDS*. In Section 4, we delve into numerical experiments. Section 5 presents the results and offers an in-depth analysis. Finally, in Section 6, we draw conclusions from our study and suggest potential avenues for future research.

## 2 Related Work

We present the related literature review by dividing this section into the following subsections: (i) NIDS, (ii) federated learning, and (iii) federated learning in NIDS.

### 2.1 NIDS

NIDS play a critical role in safeguarding computer networks from unauthorized access and malicious activities. The entire NIDS literature can be broadly categorized into signature-based and anomaly-based NIDS. A signature-based NIDS relies on the matching of seen attack signatures. An anomaly-based NIDS tries to model the expected user behavior to detect any possible deviation from the normal user behavior. With advancements in computational technology and the availability of faster computational resources anomaly-based NIDS have gained more popularity recently. Anomaly-based NIDS rely on DL and statistical learning methods to identify malicious activities [34]. To enhance the effectiveness of NIDS, the integration of DNNs has emerged as a powerful approach. DNNs excel in capturing intricate relationships and complex patterns within network traffic data [22]. Their inherent ability to learn hierarchical features makes them well-suited for discerning subtle deviations in network behavior that may signify malicious activity [53]. Additionally, DNNs can dynamically adapt to evolving attack patterns by fine-tuning and retraining on new data, ensuring persistent accuracy in detecting emerging threats [60]. Their ability to model non-linear relationships proves especially advantageous for capturing intricate attack behaviors that could evade traditional methods [27]. By capitalizing on these strengths, DNNs offer a promising avenue to elevate the accuracy and robustness of anomaly detection within packet-based NIDS.

Depending on the training data required to train such anomaly-based NIDS, there exist two prominent approaches in NIDS design: flow-based and packet-based methods. We provide insights into these approaches, highlighting their strengths, weaknesses, and recent advancements. Flow-based NIDS operate by analyzing network traffic at the flow level, where a flow represents a sequence of packets between two network endpoints. These systems focus on extracting features from flow data to identify anomalies and attacks [40, 48, 50]. Flow-based techniques have seen advancements, such as the incorporation of application metadata, which enhances their utility in network security analysis. However, flow-based NIDS may face challenges in detecting certain types of attacks due to the aggregated nature of flow data [5]. Packet-based NIDS, on the other hand, inspect individual packets in the network traffic. They analyze packet contents, headers, and payloads to detect malicious patterns [11, 15]. Packet-based analysis allows for more precise and timely detection of security incidents, enabling in-depth packet inspection and effective content-based matching against known attacks [29, 38]. Furthermore,

using raw packet features as training data alleviates the generalizability issue associated with flow-based NIDS relying on different Netflow tools [24]. Some of the advantages of using packet-based NIDS over flow-based are as follows: (1) Packet-based NIDS can perform in-depth packet inspection, allowing them to detect attacks with high precision [29]. (2) Flow-based NIDS provide a coarser view of network traffic, which makes it challenging to detect certain types of attack [5]. (3) Packet-based NIDS excel at content-based matching, making them very effective against known attacks [38]. (4) Flow-based NIDS rely on different Netflow tools (such as Wireshark and **Canadian Institute of Cybersecurity (CIC)** flowmeter) for compiling network packet information to obtain features for training, causing a loss of generalizability across different end users. Using raw packet features as training data alleviates the generalizability issue [24]. (5) Packet-based features enable real-time detection since flow-based features are obtained by analyzing the network communication after the flows are completed [18, 23].

## 2.2 Federated Learning

Federated learning has gained substantial attention as a transformative approach to decentralized **machine learning (ML)**. It enables model training across distributed data sources while preserving privacy and data locality. We provide an overview of key concepts, recent advancements, and notable research papers in the field of federated learning. McMahan et al. [36] define federated learning, in their seminal paper "Communication-Efficient Learning of Deep Networks from Decentralized Data," as an ML framework for training models across decentralized edge devices while keeping data local. This approach has evolved from the necessity to address privacy concerns and the increasing prevalence of edge and IoT devices. Federated learning typically consists of three main components: client devices (edge nodes), a central server, and federated learning algorithms. The client devices hold local data and perform local model updates, while the central server coordinates model aggregation and global updates [13]. The **federated averaging (FedAvg)** algorithm, introduced by McMahan et al. [36], is a fundamental algorithm that aggregates model updates from client devices to construct a global model. It involves iterative rounds of communication between clients and the central server. To ensure privacy, differential privacy mechanisms have been integrated into the federated learning algorithm. For example, techniques like DP-SGD [2] and secure aggregation [13] protect user data from exposure during model updates. Federated learning often assumes that client data distributions are **independent and identically distributed (IID)**. Research has focused on addressing the challenges posed by non-IID data, such as skewed distributions and non-uniform data access patterns among others [32]. Li et al. [33] modified the FedAvg algorithm to address non-IID data challenges by extending the standard federated optimization objective by adding a proximal/regularization term to the loss function. The algorithm is popularly known as FedProx [33], which encourages model updates from clients to be close to their previous models by effectively penalizing larger updates. The scope of federated learning's applications has transcended traditional boundaries, extending to domains like healthcare [59] and the IoT [57], accentuating its relevance for NIDS as well [31, 44].

## 2.3 Federated Learning in NIDS

Next, we discuss the use of federated learning in NIDS. In Table 1, we highlight some of the most recent and notable works employing federated learning in the NIDS domain. Agarwal et al. [3] provide an extensive and exhaustive review of the use of federated learning in NIDS. They also discussed the need for a federated learning paradigm, various types of intrusion detectors, and relevant ML approaches, along with potential issues. We observe that a major part of the literature focuses on building a federated learning framework for flow-based NIDS [6, 7, 9, 14, 25, 41, 49]. One of the major challenges in implementing federated learning for NIDS is data dimensionality as extracting flow-based features from different source datasets results in an inconsistent set of features across different clients. Packet-based implementation of the federated learning framework alleviates that problem since raw packet data features can be used directly for training, which remain consistent across the clients, irrespective of the source datasets. There exist very few works in the literature that employ a packet-based

Table 1. A Brief Literature Review of Federated Learning in NIDS

| Paper Name | Authors | Year | Algorithm | Global/ Local Model | Data Representation | Dataset(s) |
|---|---|---|---|---|---|---|
| Fed-ANIDS: Federated Learning for Anomaly-Based NIDS [25] | Idrissi et al. | 2023 | FedProx | Autoencoders | Flow-based | USTC-TFC2016, CIC-IDS2017, CIC-IDS2018 |
| Data-Efficient, Federated Learning for Raw Network Traffic [56] | Willeke et al. | 2023 | FedAvg | 1dCNN, FcNN | Packet-based | UNSW-NB15, CIC-IDS2017 |
| GowFed A Novel Federated Network Intrusion Detection System [9] | Belenguer et al. | 2023 | Fedavg with Gower distance | DNN | Flow-based | TON_IOT |
| Enhancing Privacy-Preserving Intrusion Detection through Federated Learning [6] | Alazab et al. | 2023 | FedAvg | DNN | Flow-based | NSL-KDD |
| Collaborative IDS for SDVN: A Fairness Federated Deep Learning Approach [14] | Cui et al. | 2023 | FedAvg with two-stage gradient optimization | 1dCNN | Flow-based | KDD99, NSL-KDD |
| A Federated Learning-Based Approach for Improving Intrusion Detection in Industrial Internet of Things Networks [42] | Rashid et al. | 2023 | FedAvg | CNN, RNN | Flow-based | Edge-IIoTset |
| FLUIDS: Federated learning with Semi-Supervised Approach for IDS [7] | Aouedi et al. | 2022 | FedAvg | Autoencoders, DNN | Flow-based | UNSW-NB15 |
| Towards Asynchronous Federated Learning-Based Threat Detection: A DC-Adam Approach [49] | Tian et al. | 2021 | DC-Adam | Denoising Autoencoders | Flow-based | CIC-IDS2017, IoT-23, MNIST |
| Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning [41] | Rahman et al. | 2020 | FedAvg | DNN | Flow-based | NSL-KDD |
| DIOT: A Federated Self-Learning Anomaly detection System for IoT [39] | Nguyen et al. | 2019 | FedAvg | GRU | Packet-based | Mirai |

NIDS with the federated learning framework. Willeke et al. [56] use raw packet features extracted from the **packet capture (PCAP)** files, while Nguyen et al. [39] consider some extracted features like packet direction, port type, and **transmission control protocol (TCP)** flags, among others. Additionally, the latter also considers a sequence of packets between the security gateway and IoT devices for the detection task. Another important challenge in implementing federated learning in NIDS is the model divergence under non-IID data scenarios. The FedAvg algorithm, used in [56], performs well under IID assumptions. However, distributed clients in the NIDS setting experience non-IID data. The FedAvg algorithm is known to have underwhelming results in the presence of non-IID data across the different clients. The DC-Adam algorithm addresses this issue by using Taylor expansion to stabilize the loss function when minimizing it along with some warm-up procedures for parameter initialization [49]. Belenguer et al. in the GowFed paper address the model divergence issue by introducing a Gower dissimilarity [19] to limit the extent of the model updates. The FedProx algorithm [33] also performs well, in general, in the presence of non-IID data. Recently, some works have also explored poisoning attacks on federated learning-based NIDS [37, 58]. However, evaluating the resilience of federated learning-based NIDS classifier systems under novel attacks such as zero-day and evasion attacks remains unexplored.

To the best of our knowledge, there is a lack of existing literature on the development of a federated learning framework for NIDS using raw packet data in a non-IID data environment. There is also a notable absence of studies that have assessed and enhanced the robustness of federated learning-based NIDS in the face of emerging

and unfamiliar attack scenarios. In this study, we aim to address these gaps by developing a federated learning framework for NIDS that is capable of recognizing known attacks and can be fine-tuned to quickly adapt to identify novel attacks across different clients.

## 3 Methodology

In this section, we present our methodology for the federated learning framework for packet-based NIDS (FedNIDS). We begin by discussing the preliminaries (Section 3.1), which include a mathematical overview of federated learning (Section 3.1.1), the non-IID data issue in federated learning (Section 3.1.2), and the problem statement (Section 3.1.3). We then introduce the proposed FedNIDS framework (Section 3.2), which consists of two main stages: federated DNN pre-training (Section 3.2.1) and federated novel attack fine-tuning (Section 3.2.2).

### 3.1 Preliminaries

*3.1.1 Federated Learning: A Mathematical Overview.* Federated learning in the client–server paradigm involves three main parties: (1) a multitude of silos (e.g., clients) denoted as $S = \{s_i\}_{i=1}^{N}$, where each $s_i$ owns private datasets $D_{s_i}$; (2) a model owner, $O$, who initiates the process with a randomly initialized global model, represented by its parameters, $w_G$, and oversees its training; and (3) a secure aggregator $Agg$ (e.g., a trusted server), positioned between $O$ and $S$. Notably, $O$ is oblivious to the specifics of $D_{s_k}$ due to the implementation of a secure aggregation protocol by $Agg$ [13]. Typically, **stochastic gradient descent (SGD)** is used by the silos to train their models (FedAvg [36]). Preceding the commencement of federated learning, $w_G$ is initialized by $O$. During the $t$th aggregation round:

(1) $Agg$ transmits $w_G^t$ to a subset of silos $S_{sub} = \{s_i\}_{i=1}^{K}$, where $K \leq N$.
(2) Each silo $s_i$ updates its local model from $w_{s_i}^{t-1}$ to $w_{s_i}^t = w_G^t$. Subsequently, $s_i$ retrains $w_{s_i}^t$ through a predetermined number of local passes over its private dataset $D_{s_i}$ and sends the retrained local model $w_{s_i}^t$ back to $Agg$.
(3) $Agg$ aggregates the retrained local models to compute a new global model $w_G^{t+1}$.

The above process continues iteratively, until the global model $w_G$ achieves the desired performance level or another predefined stopping criterion is met. The objective is to improve the global model's accuracy over successive rounds, while respecting privacy and security constraints.

*3.1.2 Non-IID Data Issue in Federated Learning.* A significant challenge within the realm of federated learning arises due to the presence of non-IID data across different silos [26, 32]. Non-IID data refer to data that do not share the same statistical characteristics among various silos. This non-IID nature of data can significantly impact the accuracy of the federated learning algorithm. The underlying reason lies in the distinctiveness between the distribution of each local dataset within different silos and the global distribution. Consequently, the objectives of the local models within each silo can be misaligned with the overarching goal of achieving global optimality. This misalignment leads to a *drift* in the local updates [28, 54, 55].

To simplify, during the local training phase within each silo, the models strive to improve their respective local objectives. However, these local objectives may lead the models to local optima that differ considerably from the desired global optima. Moreover, when substantial local updates occur, such as an extensive number of local training epochs [33], the average model derived from combining these local models can also deviate significantly from the global optima. As a result, the final global model's accuracy is notably compromised, particularly when compared to scenarios where data is IID. This challenge is visually illustrated in Figure 1, which vividly captures the hurdle faced by FedAvg when dealing with non-IID data scenarios involving silos. To elaborate further, consider the scenario where the global optimum $w_G^*$ is close to the local optima $w_1^*$ and $w_2^*$ under the IID setting. This results in the averaged model $w^{t+1}$ being close to the global optimum. However, in non-IID settings, when $w_G^*$ is distant from $w_1^*$, the averaged model $w^{t+1}$ can deviate significantly from the global

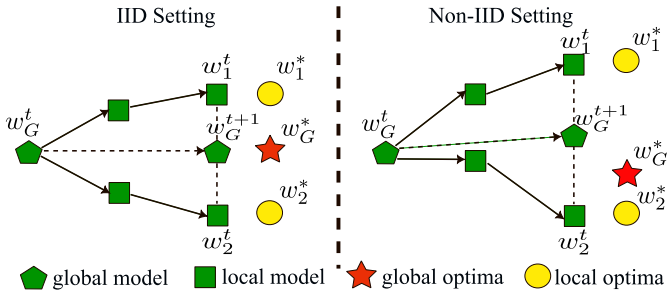


Fig. 1. Example of a drift under the non-IID setting.

optimum. Effectively tackling this challenge requires designing robust algorithms tailored to handle non-IID data present across different silos.

*3.1.3 Problem Statement.* Consider $N$ clients or silos denoted as $S = \{s_i\}_{i=1}^{N}$, each possessing a local dataset $D_{s_i}$ containing network traffic samples. The objective is to develop a global classification model that can accurately identify malicious samples within the network traffic data from these distributed datasets, while upholding data privacy and security. While flow-based analysis has its merits in providing a high-level overview of network activity, our study focuses on packet-based analysis to address the challenges of non-IID data in a federated learning setting for NIDS. We pose the problem as follows: Given the distributed datasets $D_{s_i}$ develop a federated learning-based DNN model that optimizes a global objective function while respecting the privacy constraints of individual clients. Mathematically, we aim to minimize the global objective function $\mathcal{L}(w)$, which is defined as

$$\arg\min_{w} \mathcal{L}(w) = \sum_{i=1}^{N} \frac{|D_{s_i}|}{|D|} \mathcal{L}_i(w).$$

In the above equations:

—$|D_{s_i}|$ represents the size (number of samples) of the local dataset for a particular silo $s_i$.
—$|D|$ represents the total size (number of samples) of the entire dataset, which comprises the combined data from all silos.
—$\mathcal{L}_i(w)$ is the local objective function for silo $s_i$, measuring the goodness of fit or the loss of the global model $w$ on the data specific to silo $s_i$. It is defined as $\mathcal{L}_i(w) = \mathbb{E}_{x;y\sim D_{s_i}}[\ell_i(w;x;y)]$.

The optimization problem, $\arg\min_w \mathcal{L}(w)$, seeks to find the global model parameters $w$ that minimize the objective function $\mathcal{L}(w)$ ensuring effective performance across all data sources while respecting privacy constraints in federated learning. This is achieved by aggregating local losses $\mathcal{L}_i(w)$ from each source, weighted by their data contributions, to formulate $\mathcal{L}(w)$. Minimizing $\mathcal{L}(w)$ aims to create a model that balances performance across all sources, promoting effectiveness on the combined data from multiple parties, the core objective in distributed data scenarios. Our research aims to develop a federated learning-based DNN model that can effectively detect attacks across different silos and efficiently adapt to novel attack patterns when new data becomes available.

## 3.2 Federated Learning Framework for Packet-Based Network Intrusion Detection System (*FedNIDS*)

To attain the aforementioned research objective, we propose a generalized federated learning framework, named *FedNIDS*, to enhance both the robustness and the performance of federated models when learning from decentralized data with statistical heterogeneity in NIDS. Our framework comprises two stages: a federated DNN pre-training stage and a federated novel attack fine-tuning stage, as shown in Figure 2 and Algorithm 1. During the federated DNN pre-training stage, the model exploits knowledge from decentralized data by pre-training with

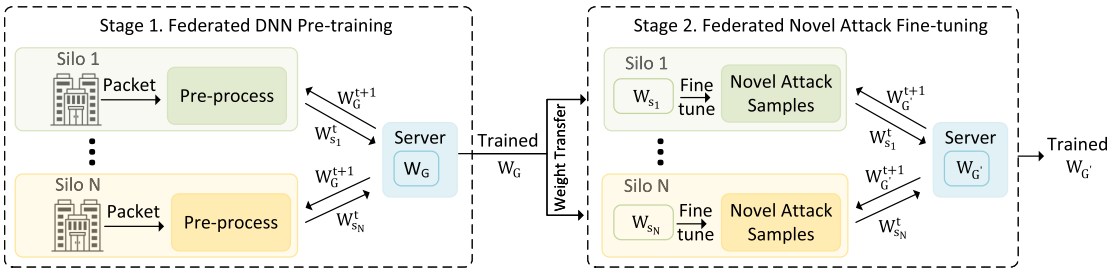


Fig. 2. Overview of the federated learning framework. In the pre-training stage (left), the packet data are processed to learn representations in each silo. The pre-training process consists of three steps and ends when it reaches the maximum communication rounds $T$. At round $t$, (1) Each silo $S_i (i \in \{1, ..., N\})$ trains its local DNN with local data; (2) Silo $s_i$ uploads the weights $w_{s_i}^t$ to the central server; (3) The server produces a global DNN model with weights $w_{s_i}^{t+1}$ via model weights averaging and broadcasts the global model back to each local silo. In the fine-tuning stage (right), the final pre-trained global $W_G$ from the first stage is used to initialize each local DNN. The $W_G$ is then fine-tuned with new attack samples. End-to-end federated fine-tuning is performed on the novel attack samples in each local silo to get the desired model $W_{G'}$.

pre-processing procedure in a distributed setting. In the federated novel attack fine-tuning stage, the knowledge is transferred from the previous stage to the target task by fine-tuning the federated models with novel attack samples.

Algorithm 1 presents the pseudocode for *FedNIDS*. The algorithm consists of two main stages: federated DNN pre-training and federated novel attack fine-tuning. In the federated DNN pre-training stage (lines 2–9), the algorithm iteratively trains the global model by aggregating the local model updates from the participating silos. The local training is performed on each silo's data (lines 12–18), where the local models are updated using their respective datasets. The model updates from each silo are then aggregated to update the global model (line 8). In the federated novel attack fine-tuning stage (lines 19–23), the algorithm adapts the pre-trained global model to detect novel attack patterns. When a silo encounters novel attack samples, the fine-tuning process is triggered. The local models are updated using these novel attack samples (lines 20–22), allowing the global model to learn and adapt to new attack patterns. In the following Sections 3.2.1 and 3.2.2, we provide detailed descriptions of federated DNNs pre-training and federated novel attack fine-tuning stages, respectively.

*3.2.1 Federated DNNs Pre-Training.* The first stage of our proposed *FedNIDS* framework focuses on enhancing the global model's initialization through federated DNN pre-training. The goal is to develop robust initialization of local DNNs at each silo that can effectively capture relevant features from their respective datasets. This stage reduces the need for extensive communication rounds between the clients and the server, thereby enhancing the efficiency of the federated learning process.

The pre-training process occurs in a distributed setting and involves the following steps, repeated for a maximum of $T$ communication rounds:

(1) Pre-Processing: Numerical features are extracted from the raw PCAP data files. The pre-processing step is kept simple and generalizable across different organizations. As discussed earlier, flow-based features raise the issue of inconsistent feature sets across organizations and hinder generalizability. Hence, we use the raw PCAP files to extract numerical features. The dataset creation tool is adopted from [23]. A comprehensive guide to extracting and labeling the PCAP files are as follows:
   —Use Scapy [12] and dpkt [47] to parse different header and segment information from the raw PCAP file.
   —Use the 6-tuple information containing source IP address, destination IP address, source port, destination port, protocol, and epoch time to differentiate between benign and malicious packets.

**Algorithm 1:** FedNIDS

**Input:** Number of communication rounds $T$, number of local epochs $E$, number of silos $N$, local data set $D_{s_i}$, learning rate $\eta$

**Output:** Global models $w_G, w_{G'}$

**Server executes:**

Initialize $w_G^0$;
**for** each round $t = 0, 1, ..., T-1$ **do**
  Sample a set of clients $S^t$;
  $n \leftarrow \Sigma_{i \in S^t} |D_{s_i}|$;
  **for** each silo $s_i \in S^t$ in parallel **do**
    Send the global model $w_G^t$ to silo $s_i$;
    $\Delta w_{s_i}^t \leftarrow$ **LocalTraining**$(s_i, w_G^t)$;
▷ *Stage 1: Federated DNN Pre-training*
  $w_G^{t+1} \leftarrow \Sigma_{i=1}^{N} \frac{|D_{s_i}|}{n} \Delta w_{s_i}^t$;
▷ *Stage 2: Federated Novel Attack Fine-tuning*
  $w_{G'}^{t+1} \leftarrow \Sigma_{i=1}^{N} \frac{|D_{s_i}|}{n} \Delta w_{s_i}^t$;

**Client executes:**

$\mathcal{L}(w; \mathbf{b}) = \sum_{(x,y) \in \mathbf{b}} \ell(w; x; y) + \frac{\mu}{2} ||w - w_t||^2$;
**LocalTraining**$(s_i, w_G^t)$;
▷ *Stage 1: Federated DNN Pre-training*
Sample batches $\mathbf{b} = \{x, y\}$ from local data $D_{s_i}$;
$\mathbf{b}' \leftarrow$ Pre-process$(\mathbf{b})$;
$w_{s_i}^t \leftarrow w_G^t$;
**for** each local epoch $k = 1, ..., E$ **do**
  **for** each batch $\mathbf{b}'$ **do**
    $w_{s_i}^t \leftarrow w_{s_i}^t - \eta \nabla \ell(w_{s_i}^t; \mathbf{b}')$;
$\Delta w_{s_i}^t \leftarrow w_G^t - w_{s_i}^t$;
▷ *Stage 2: Federated Novel Attack Fine-tuning*
**for** each local epoch $k = 1, ..., E$ **do**
  **for** each batch $\mathbf{b}'$ **do** ▷ Updated batch $\mathbf{b}'$ with novel attack samples
    $w_{s_i}^t \leftarrow w_{s_i}^t - \eta \nabla \ell(w_{s_i}^t; \mathbf{b}')$;
$\Delta w_{s_i}^t \leftarrow w_{G'}^t - w_{s_i}^t$;
return $\Delta w_{s_i}^t$ to the Server;

—Extract only forward packets resulting in a dataset of unidirectional communication, i.e., only extract packets that are being sent from identified source IP addresses excluding any response from the server.
—To alleviate inherent bias in training ML models, we remove some header information (ethernet header, source and destination IP address, source and destination port information).
—Set a maximum feature length for all extracted packets. However, the number of bytes in a packet may vary greatly. To counter this zero-padding is used that helps in maintaining a standard structure of the data.

—Convert raw packet information from hexadecimal to decimal. Each byte value contains a combination of two hexadecimal numbers, ranging from 00 to $FF$. The hexadecimal numbers are converted to decimal numbers between 0 and 255 and subsequently normalized for efficient machine computation.

(2) Each silo $s_i$ (where $i \in \{1, \ldots, N\}$) trains its local DNN-based classifier with its own pre-processed data for $E$ local epochs.
(3) Silo $s_i$ uploads the weights $w_{s_i}^t$ of its DNN to the central server.
(4) The central server aggregates the uploaded weights using a model weights averaging technique and produces an updated global DNN model with weights $w^{t+1}$.
(5) The server broadcasts the updated global model to each local silo $s_i$.

In addition to the pre-training stage, our *FedNIDS* framework incorporates the FedProx technique [33] to enhance the local objective function based on the FedAvg algorithm. This technique introduces an $L2$ regularization term in the local objective function to limit the distance between the local model and the global model. This regularization term effectively controls the size of local updates, preventing the local models from diverging too far from the global optima. The effectiveness of the regularization term in handling non-IID data has been demonstrated in various studies [28, 33]. These studies have shown that FedProx achieves better performance and faster convergence compared to FedAvg in the presence of non-IID data. A hyper-parameter $\mu$ is introduced to adjust the weight of the $L2$ regularization term. While the modification to FedAvg is lightweight and easy to implement, it introduces some additional computation overhead. However, it does not introduce extra communication overhead. This approach requires fine-tuning of $\mu$ to achieve superior accuracy. If $\mu$ is too small, the regularization term might have minimal impact. Conversely, if $\mu$ is too large, the local updates become very small, potentially leading to slower convergence speed.

This stage concludes when the maximum communication rounds $T$ are reached. The global model $w_{T+1}$ obtained at the end of the pre-training stage serves as a robust initialization for the subsequent novel attack fine-tuning stage. The combination of pre-processing and federated DNN pre-training enhances the quality of the learned features and the global model's ability to capture important characteristics of network traffic data, improving the overall effectiveness of the *FedNIDS* framework.

*3.2.2 Federated Novel Attack Fine-Tuning.* In the second stage of our FedNIDS framework, we perform federated novel attack fine-tuning by utilizing the knowledge gained during the pre-training stage to the target task of network intrusion detection. It is important to note that the fine-tuning step differs from periodic retraining of the global model. Periodic retraining involves regularly updating the model with new data from the participating silos to maintain its performance and relevance over time. This process typically includes both benign and malicious traffic patterns and is performed at regular intervals. In contrast, the fine-tuning step specifically focuses on adapting the model to novel attack patterns that were not present in the initial training data. It allows for a rapid response to emerging threats by quickly updating the model with novel attack samples. The fine-tuning step can be triggered whenever novel attack patterns are detected, while periodic retraining can be performed at regular intervals to incorporate a broader set of newly observed data. The fine-tuning step provides an additional layer of adaptability to address novel attack patterns in a timely manner.

The fine-tuning process occurs as follows:

(1) The trained global model $w_{T+1}$ from the pre-training stage is used to initialize each local DNN at the respective silos.
(2) Each silo $s_i$ continues to contribute to the learning of the global model with their own local datasets $D_{s_i}$, which include newly experienced novel attack samples.
(3) The fine-tuning involves updating the DNN's parameters, $w$, using an optimization algorithm such as SGD to minimize the loss on the novel attack samples.

This federated novel attack fine-tuning stage effectively adapts the trained global model to the specific characteristics of each silo's dataset, enabling the model to detect new attack patterns that may arise in the future. The combination of the two stages within the *FedNIDS* framework aims to enhance the model's accuracy, robustness, and generalization capabilities for network intrusion detection.

## 4 Experiments

In this section, we outline the numerical experiments performed to evaluate our *FedNIDS* methodology. We first discuss the experimental data followed by the description related to the non-IID data silo creation. Subsequently, we discuss the hyper-parameter configurations employed for *FedNIDS*. We evaluate the performance of our approach by comparing it against two other federated learning approaches from the literature: (1) FedAvg [56], which was recently utilized for NIDS and (2) FedAdam [43], an extension of the Adam optimizer to the federated setting, which uses moment estimates to adapt the learning rates for each model parameter. The experimentation was carried out employing a computing system featuring the NVIDIA GeForce RTX 3070 graphics processing unit with 12 GB of video memory and a substantial 128 GB of DDR4 RAM operating at a speed of 3,200 MHz. We built a Python-based simulator to emulate the federated learning process, in which the total available computational resources were equally divided amongst the silos.[1]

### 4.1 Data Description

A federated learning framework in the context of NIDS offers significant advantages in preserving both data privacy and security, all the while maintaining data ownership and control. Nevertheless, obtaining real-world organizational traffic data for research is a daunting task for the same reasons. Hence, researchers turn to publicly available benchmark datasets as a practical substitute. In this study, we use raw PCAP files from two publicly available intrusion detection datasets, namely, CIC-IDS2017 [45] and CIC-IDS2018 [46] to extract benign and attack communications. We select CICIDS datasets over some popular yet older datasets like NSL-KDD and KDD-CUP due to the following reasons: (i) they provide a more pragmatic representation of modern network traffic [21] and (ii) they provide easy accessibility to the raw PCAP files, thereby reducing the dependency on flow-level data. These datasets are accumulations of network activities recorded over several days. The details can be found on the CIC Web site [1], and the threat model for the conducted attacks is outlined in [46]. A brief description of the PCAP files, in terms of their size and contents, is provided in Table 2.

We follow the process in [23] to obtain the packet information from the PCAP files. We analyze both the header and payload sections of the packet. The header section includes information such as **time to live (TTL)**, window size, fragmentation, total length, acknowledgment number, and flags, among others. The payload section contains the actual data being communicated. These features can provide insights into various attacks. For example, unusually large packet sizes may indicate attempts to exploit vulnerabilities by overflowing buffers or injecting malicious payloads. Additionally, anomalies in packet headers, such as forged or spoofed TTL values, can indicate attempts to evade detection.

We exclude Botnet attack packets from the CIC-IDS2017 dataset during the initial phase of the experimentation so that we can introduce Botnet attack as a zero-day attack to evaluate the resilience of our framework. In our experimentation, we focus on analyzing unidirectional forward packets, which are packets sent from source IP addresses, excluding any responses from the server. This approach allows us to concentrate on detecting attacks originating from specific sources, enabling effective attribution and facilitating targeted security measures. By considering only the forward packets, as in [23], simplify the data pre-processing and feature extraction process, focusing on the relevant information carried in the packets sent by potential attackers.

In a TCP packet, the maximum byte count is 1,554, distributed as follows: 1,460 bytes for the application layer, 60 bytes for the transport layer (20 for the header and a maximum of 40 for options), 20 bytes for the internet

[1]We have made the source code accessible at the following URL: https://github.com/quocnh/Fed_NIDS

Table 2. Description of CIC-IDS2017 and CIC-IDS2018 PCAP Files

| CIC-IDS2017 | | | CIC-IDS2018 | | |
|---|---|---|---|---|---|
| **Day/Date** | **File Size** | **Activity** | **Day/Date** | **File Size** | **Activity** |
| Monday/ 3 July 2017 | 10 GB | Benign | Wednesday/ 14 February 2018 | 147 GB | SSH/FTP Patator and Benign |
| Tuesday/ 4 July 2017 | 10 GB | Brute Force, and Benign | Thursday/ 15 February 2018 | 57.8 GB | DoS Goldeneye, DoS Slowloris, and Benign |
| Wednesday/ 5 July 2017 | 12 GB | DoS, and Benign | Friday/ 16 February 2018 | 460 GB | DoS Slowhttptest, DoS Hulk, and Benign |
| Thursday/ 6 July 2017 | 7.7 GB | Web Attack, Infiltration, and Benign | Wednesday/ 21 February 2018 | 97.5 GB | DDoS and Benign |
| Friday/ 4 July 2017 | 8.2 GB | Botnet, Port Scan, DDoS, and Benign | Thursday/ 22 February 2018 | 110 GB | Web Attack and Benign |
| - | - | - | Friday/ 23 February 2018 | 65.9 GB | Web Attack and Benign |
| - | - | - | Wednesday/ 28 February 2018 | 73.1 GB | Infiltration and Benign |

layer, and 14 bytes for the Ethernet layer. To prevent potential bias in the learning process of DL models, certain information is excluded, such as MAC addresses, as well as source and destination IP addresses. Consequently, we are left with 1, 525 features, all of which are kept in our dataset to ensure segment information is not excluded. The hex payload values in these features are converted to decimal values between 0 and 255, and later normalized to values between 0 and 1.

To evaluate the robustness of the *FedNIDS* framework, we also subject it to adversarial examples generated by modifying different attack packets encountered in the CICIDS datasets. We created two different toolchains to generate these evasion attacks with both constrained and unconstrained perturbations. Toolchain 1 was developed using a DRL approach from literature [23], which makes subtle perturbations such that the functionality and maliciousness of these packets remain intact (constrained). Table 3 shows the features that we considered for perturbations using Toolchain 1 and their resulting impact on other features. Toolchain 2 was developed using a heuristic approach, in which we made perturbations by making a random selection of features (unconstrained). We then methodically increased the feature values by a constant value as in [18]. We considered a white-box setting for the generation of the adversarial samples to test the worst-case scenario for the NIDS. In this setup, we considered a DNN model with the same architecture as used by the global model in the federated learning framework. The samples that evaded the DNN model were collected and stored for each of these toolchains.

## 4.2 Experimental Setup

Next, we discuss the segregation of the data into four different silos to evaluate the effectiveness of the *FedNIDS* framework. In the FL-NIDS literature, most of the research studies focus on creating a global model with the IID assumption. In a realistic scenario, where we want to create a global NIDS with data coming from different organizations, often it is very difficult to adhere to the IID assumption. To counter this, we partition the

Table 3. Description of Perturbations Used to Generate Adversarial Examples

| Perturbation | Description | Impact |
|---|---|---|
| Modifying fragmentation | Fragmentation bytes can be modified from do not fragment to do fragment/more fragment or vice versa. | Directly impacts fragmentation bytes (byte numbers 7, 8 of the IP header) indirectly impacts TTL value (byte number 9 of the IP header) and IP checksum (byte numbers 11, 12 of the IP header). |
| Modifying TTL | Increasing/decreasing the TTL value by a small integer between 1 and 255. | Directly impacts the TTL byte (byte number 9 of the IP header) and indirectly impacts IP checksum (byte numbers 11, 12 of the IP header). |
| Modifying window size | Increasing/decreasing the window size value by a small integer between 1 and 65335. | Directly impacts window size bytes (byte number 15, 16 of the TCP header) and indirectly impacts TCP checksum (byte numbers 17, 18 of the TCP header). |
| Adding/modifying maximum segment size | Only limited to SYN and SYN-ACK packets. The MSS value is limited between 0 and 65335. | Directly impacts TCP options (bytes after 20 of the TCP header) and indirectly impacts TCP checksum (byte numbers 17, 18 of the TCP header). |
| Adding/modifying window scale value | Only limited to SYN and SYN-ACK packets. The MSS value is limited between 0 and 14. | Directly impacts TCP options (bytes after 20 of the TCP header) and indirectly impacts TCP checksum (byte numbers 17, 18 of the TCP header). |
| Adding segment information | Add dead bytes/most commonly occurring segment information. | Directly impacts the TCP segment, and indirectly impacts TCP checksum (byte numbers 17, 18 of the TCP header). |

extracted data based on the shards partition used in [36], a popular method for segregating data into non-IID partitions/shards. There exist multiple strategies for shard partitioning, namely, range-based, hash-based, key-based, and round-robin. In this research, we employ hash-based shard partitioning, which distributes data across shards based on a hash function applied to a specific attribute. The selected attribute is the label of the packet data. We created silos based on the selected partitioning strategy. Table 4 illustrates how the various malicious and benign samples from the two datasets are segregated into different silos. To maintain data integrity and ensure the reliability of our model's performance, we partition the data within each silo into separate training, validation, and testing sets. The training set is utilized to update the model parameters during the federated learning process, while the validation set is used for hyper-parameter tuning and model selection. The testing set is kept entirely separate and is exclusively used for the final evaluation of the trained model. This stringent data segregation protocol ensures that there is no overlap between the data subsets, preventing any information leakage that could lead to overly optimistic performance estimates.

The silos are carefully constructed to preserve the non-IID nature of the data across them and to demonstrate the attack detection efficacy of our framework without explicit data sharing. It is also important to note that some silos experience attacks that no other silos encounter in this setup. As discussed in the previous sections, having non-IID data across different silos poses a major challenge to the traditional federated learning frameworks, which our methodology aims to overcome. To exhibit the non-IID nature of the partitioned silos, we perform pairwise hypothesis testing. Particularly, to assess whether two samples have the same underlying distribution,

Table 4. Data Description for Different Silos

| Silo | Packet Type | Source Data set | Number of Forward Packets | | |
|---|---|---|---|---|---|
| | | | Train | Validation | Test |
| Silo 1 | Port Scan, DoS, DDoS, and Benign | CIC-IDS2017 | 133,906 | 15,000 | 33,476 |
| Silo 2 | Brute Force, Infiltration, Web Attack, and Benign | CIC-IDS2017 | 110,585 | 15,000 | 30,146 |
| Silo 3 | DoS, Brute Force, and Benign | CIC-IDS2018 | 102,947 | 15,000 | 25,736 |
| Silo 4 | DDoS, Infiltration, Web Attack, and Benign | CIC-IDS2018 | 102,324 | 15,000 | 25,581 |

Table 5. Percentage of Non-IID Samples between Different Silos

| Silo | Silo 1 | | Silo 2 | | Silo 3 | | Silo 4 | |
|---|---|---|---|---|---|---|---|---|
| Statistical Tests | Rank Sum Test | K-S Test | Rank Sum Test | K-S Test | Rank Sum Test | K-S Test | Rank Sum Test | K-S Test |
| Silo 1 | - | - | 65.02 | 90.80 | 90.52 | 91.54 | 93.62 | 94.45 |
| Silo 2 | 65.02 | 90.80 | - | - | 90.22 | 98.57 | 93.20 | 97.84 |
| Silo 3 | 90.52 | 91.54 | 90.22 | 98.57 | - | - | 87.63 | 88.14 |
| Silo 4 | 93.62 | 94.45 | 93.20 | 97.84 | 87.63 | 88.14 | - | - |

we perform the **Kolmogorov-Smirnov (K-S)** test [10]. We also perform the Mann-Whitney rank sum test [35] to compare two independent samples to determine if they have different distributions, or if one tends to have larger values than the other. Table 5 exhibits the pairwise K-S test and rank sum test results among different silos of data. We perform the tests on a small subset of 500 random samples drawn from each silo of packet data that has some payload information. Each sample from each silo is compared against all other samples from all other silos. The numbers reported in Table 5 are the percentage of times the K-S test/the rank sum test result indicated toward two samples being different. The high percentages reported in Table 5 indicate a significant difference between the data distributions of the silos. The percentage values represent the proportion of pairwise comparisons where the statistical tests (K-S test and rank sum test) rejected the null hypothesis of identical distributions. Higher percentage values suggest a greater degree of non-IID nature between the silos. In our experiments, we were able to reject the null hypothesis of the rank sum test on an average 86.70% of the total conducted experiments and the K-S test null hypothesis got rejected 93.55% of the total conducted experiments. From Tables 4 and 5, we can determine that the data partitions we crafted are of non-IID nature.

To demonstrate the effectiveness of the federated learning framework, we consider a setup with four clients, denoted as $N = 4$ silos, actively participating in training the global model during each communication round. Creating the training dataset for each silo poses a significant challenge due to the inherent imbalance across various attack and benign packet types. To address this, we employed a strategy wherein, for each silo, a subset of attack packets was randomly selected from the available attack types. Concurrently, we downsized the pool of benign packet samples to equal the total number of attack samples. This procedure was replicated across all silos in the experiment. For example, in Silo 1, the abundance of DDoS packets far outweighs that of other attack types, while DoS attack packets are relatively scarce. To rectify this, we randomly selected a subset of DDoS

Table 6. Hyper-Parameter Values for the DNN Model

| **Hyper-Parameter** | **Grid Search** | **Optimal Value** |
|---|---|---|
| Regularization ($\mu$) | 0.001, 0.005, 0.01, 0.05, 0.1 | 0.005 |
| Dropout rate | 0.1, 0.2, 0.3, 0.4, 0.5 | 0.3 |
| Architecture | [512, 256, 128, 64], [256, 128, 64], [128, 64] | [512, 256, 128, 64] |
| Activation function | ReLU, Tanh, Sigmoid | ReLU |
| Learning rate ($\eta$) | 0.1, 0.01, 0.001, 0.0001 | 0.01 |
| Batch size ($b$) | 32, 64, 128, 256 | 32 |
| Epochs ($E$) | 5, 10, 15, 20 | 5 |

and Port Scan packets, ensuring they match the number of DoS packets. Subsequently, we balanced the benign packets by randomly downsampling them to align with the total number of attack packets.

The choice of hyper-parameters plays a crucial role in achieving optimal performance. The architecture and hyper-parameter values were selected based on literature studies [9, 41], experimental results, and computational constraints. It is important to note that the hardware and GPU cards can significantly influence the search for optimal hyper-parameters. With additional computational resources, a more extensive grid search, incorporating larger grid sizes and a greater number of hyper-parameters, could be conducted. The tradeoff lies in the compute time required for such a comprehensive search. In our experiments using the system described earlier in this section, we employed grid search cross-validation to identify the best hyper-parameters. The maximum grid size across all tuned hyper-parameters was five.

Techniques such as L2 regularization, dropout, early stopping, and architecture search were employed to avoid overfitting. Table 6 displays the grid of hyper-parameter values searched, along with the optimal values selected for the DNN model. The following parameter values were chosen based on extensive experimentation: number of rounds ($T = 5$) governs the frequency of communication rounds between clients and the central server, allowing for iterative updates; number of local epochs ($E = 5$) dictates the number of times each client refines its local model on its respective dataset, influencing the granularity of updates; a batch size of 32 samples ($b = 32$) determines the number of data samples processed together in each iteration, balancing parallelization and memory constraints; learning rate ($\eta = 0.01$) regulates the step size of parameter updates, influencing the convergence speed; and the regularization parameter ($\mu = 0.005$) helps mitigate overfitting by adding a penalty term based on the model weights. These values were meticulously chosen to strike a balance between convergence speed, model generalization, and available computational resources. In addition to hyper-parameters, we tailored the architecture of our DNN to best suit our objectives. The *input_size* was set to 1,525, reflecting the number of features in the dataset and defining the input data's dimensionality. The network architecture encompassed four hidden layers with sizes of 512, 256, 128, and 64 units respectively, enabling the model to progressively learn abstract features from the input data. The activation function was **rectified linear unit (ReLU)**. The *num_classes* was set to 2, signifying the model's task of predicting between two output classes: benign or malicious network traffic. Consistency in architecture and hyper-parameter values was maintained across all algorithms considered for comparison.

## 5 Results

In this section, we present the experimental results and conduct an analysis of them. First, we delve into the performance of the global *FedNIDS* model against the various types of attacks observed at different silos. Next, we assess the framework's adaptability to novel attacks, encompassing previously unseen attack types and adversarially perturbed evasion attacks.

Table 7. Performance Comparison of FedNIDS with Other Approaches

| Silo | Class | FedAvg | | | FedAdam | | | Centralized | | | FedNIDS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Prec | Rec | F1 | Prec | Rec | F1 | Prec | Rec | F1 | Prec | Rec | F1 |
| Silo 1 | Attack | 1.00 | 0.97 | 0.98 | 0.99 | 0.99 | 0.99 | 0.99 | 0.98 | 0.99 | **1.00** | **0.99** | **0.99** |
| | Benign | 0.95 | 1.00 | 0.98 | 1.00 | 1.00 | 1.00 | 0.97 | 0.99 | 0.98 | 1.00 | 1.00 | 1.00 |
| Silo 2 | Attack | 1.00 | 0.96 | 0.98 | 0.99 | 0.99 | 0.99 | 0.99 | 1.00 | 0.99 | **1.00** | **0.99** | **0.99** |
| | Benign | 0.95 | 1.00 | 0.97 | 1.00 | 1.00 | 1.00 | 0.99 | 1.00 | 0.99 | 0.96 | 0.93 | 0.95 |
| Silo 3 | Attack | 1.00 | 0.68 | 0.80 | 0.99 | 0.65 | 0.78 | 0.99 | 0.90 | 0.95 | **1.00** | **0.95** | **0.96** |
| | Benign | 0.75 | 1.00 | 0.86 | 0.74 | 1.00 | 0.85 | 0.91 | 0.99 | 0.95 | 0.97 | 1.00 | 0.96 |
| Silo 4 | Attack | 1.00 | 0.67 | 0.80 | 1.00 | 0.64 | 0.78 | 0.99 | 0.90 | 0.94 | **1.00** | **0.96** | **0.94** |
| | Benign | 0.75 | 1.00 | 0.86 | 0.73 | 1.00 | 0.84 | 0.90 | 0.99 | 0.94 | 0.98 | 0.97 | 0.95 |

The bold values represent top scores.

### 5.1 Performance of the Global FedNIDS Model

We evaluate the performance of the trained global *FedNIDS* model at each silo and compare it to the FedAvg algorithm [56], the FedAdam algorithm [43], and a centralized approach, where the same DNN model is trained using the combined data from all four silos. In the centralized approach, we train the model by consolidating the training sets from each of the silos and evaluating it on the same testing data samples as the other approaches. Table 7 presents precision (prec), recall (rec), and F1 score metrics for both attack and benign classes. We report the performance of each approach in accurately classifying attack and benign samples for each silo. Notably, the global *FedNIDS* model exhibits the strongest performance across all the silos and among all the methods, achieving the highest precision and recall values for attack samples.

For Silos 1 and 2, *FedNIDS* demonstrates remarkable precision, recall, and F1 score values for both attack and benign classes. Specifically, for the attack class, the model achieves precision, recall, and F1 score values greater than or equal to 0.99, indicating its ability to accurately identify attack samples with minimal false positives or false negatives. Similarly, for the benign class, the model exhibits high precision and recall values, resulting in F1 scores of 1.00 and 0.95, respectively. This showcases the model's ability to maintain accurate classification of benign samples, while effectively detecting attacks.

In Silos 3 and 4, where attack patterns are more complex and diverse, the global *FedNIDS* model maintains strong performance. For the attack class in Silos 3 and 4, the model achieves precision values of 1.00 and recall values greater than or equal to 0.95, with F1 scores at or above 0.95. This reflects the model's ability to effectively classify various attack samples. Furthermore, in Silo 3, the model demonstrates high precision (0.97) and recall (1.00) for the benign class, resulting in an F1 score of 0.96. In Silo 4, the model achieves a precision of 0.98 and a recall of 0.97 for the benign class, yielding an F1 score of 0.95. These outcomes underline the robustness and adaptability of the global *FedNIDS* model to different types of samples in the silos' datasets.

An intriguing observation is that *FedNIDS* attains a similar level of performance as the centralized approach and outperforms other federated learning algorithms, such as FedAvg and FedAdam. The results in Table 7 show that *FedNIDS* surpasses FedAvg's performance, particularly in Silos 3 and 4, where the attack patterns are more diverse. The superior performance of *FedNIDS* compared to FedAvg can be attributed to its ability to handle non-IID data distributions more effectively. *FedNIDS* adapts to the unique characteristics of each silo's network environment and captures the intricate patterns in the network traffic data. On the other hand, FedAdam demonstrates comparable performance to *FedNIDS* in Silos 1 and 2 but falls short in Silos 3 and 4. The adaptive moment estimation used in FedAdam helps in accelerating the convergence and improving the model's ability

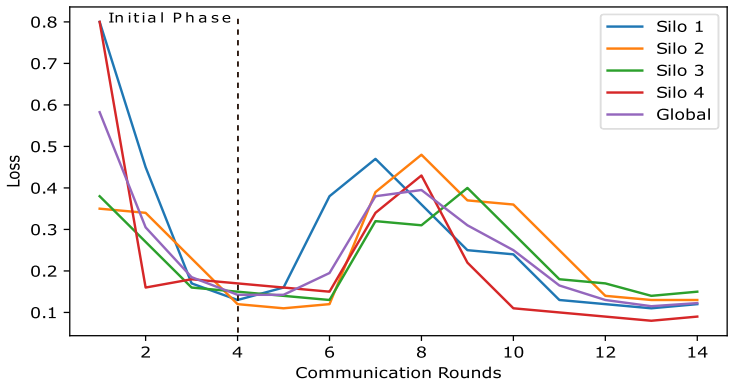


Fig. 3. FedNIDS adaptation to emerging threats.

to handle non-IID data to some extent. However, the results suggest that FedAdam may not be as effective as *FedNIDS* in capturing the complex attack patterns in Silos 3 and 4, where the attack samples are more diverse.

The superior performance of *FedNIDS* compared to FedAvg and FedAdam highlights the potential of our proposed framework in achieving accurate and robust detection outcomes in a federated learning setting. This comparative analysis underscores the importance of selecting appropriate federated learning algorithms and techniques tailored to the specific requirements of the NIDS domain. The ability of *FedNIDS* to maintain high performance across all silos, surpassing the centralized approach and other federated learning algorithms, demonstrates its potential for real-world deployment in decentralized network environments. By leveraging the strengths of federated learning while addressing the limitations of existing algorithms, *FedNIDS* offers a promising solution for accurate and responsive network intrusion detection in a privacy-preserving manner. Figure 3 further illustrates the convergence of the loss values at all the silos by the end of the fourth round, indicating the effectiveness of *FedNIDS* in learning from distributed data and adapting to the unique threat landscapes of each silo.

## 5.2 Adaptation to New Threats

Network security is a dynamic field, demanding adaptive and robust solutions to counter emerging threats. In this experiment, we evaluate the capability of *FedNIDS* to adapt to new, previously unseen attacks. To perform this evaluation, we introduce held-out botnet attack samples, which had not been encountered by any of the silos previously. Silo 1 is exposed to these novel attack samples. It is important to note that, at the time of this experiment, all silos were employing the trained global model, whose performance had been detailed in the previous experiment after four rounds of communication (refer to Figure 3).

This experiment aims to demonstrate the efficacy of the second stage of the *FedNIDS* framework. Figure 3 (after round four) illustrates the subsequent rounds of updates between the clients and the server (as outlined in Algorithm 1), starting from the moment Silo 1 encountered the botnet attack samples. Initially, it is evident that the trained global model struggles with detecting the botnet attack pattern from Silo 1. This challenge is reflected in the higher loss values, indicating suboptimal performance during the early rounds. Nevertheless, as the training progresses with additional rounds, the global model gradually learns to recognize the botnet attack pattern through the weight updates originating from the Silo 1 model.

After approximately four additional rounds, the model exhibits a significant reduction in loss across all silos, signifying a successful adaptation to the detection of botnet attack samples. The global model curve, which represents the average performance of all silos, illustrates the collective learning process. We conducted multiple trials involving various previously unseen attacks from the CICIDS datasets and consistently observed that

*FedNIDS* could adapt to new threats within approximately four rounds, on average. This experiment effectively demonstrates that the *FedNIDS* framework can adeptly respond to new threats, providing an efficient solution for sharing knowledge (rather than raw data) about a new attack experienced at one silo with all the participants.

For a real-world implementation, it is crucial to take into account the frequency of communication rounds between the clients and the central server. Depending on the unique requirements and threat landscape at each silo, the update intervals may differ. Some silos might necessitate more frequent updates, perhaps on a daily, weekly, or monthly basis, to guarantee timely adaptation to emerging threats. This adaptive scheduling of communication rounds can be customized to suit the specific needs of individual silos, enabling a finely tuned response to evolving security challenges.

### 5.3 Evaluating the Resilience of FedNIDS against Adversarial Attacks

The next experiment is designed to provide a comprehensive assessment of the resilience of *FedNIDS* against adversarial samples at two critical stages: immediately after the initial training (Stage 1) and following fine-tuning with additional adversarial data (Stage 2). The results from this experiment offer crucial insights into how *FedNIDS* adapts to adversarial data and whether the fine-tuning process, aided by an increased quantity of adversarial samples, enhances its effectiveness.

The evaluation process was conducted as follows: We employed adversarial datasets generated by the two toolchains, as elaborated in the previous section (Section 4), for this assessment. These datasets were created to mimic potential real-world evasion attacks. During Stage 2, we divided the total number of samples that successfully evaded the DNN model using these toolchains into testing and training sets. In Stage 1, we assessed the *FedNIDS* global model using the 20% testing set of adversarial samples, which was unique to each silo. In Stage 2, the *FedNIDS* model underwent a fine-tuning process using the additional 80% of adversarial training data, which was combined with the original training samples at each silo. Subsequently, we evaluated the proposed algorithm again on the same 20% testing set of adversarial samples.

The results, as illustrated in Figure 4, demonstrate that during Stage 1, the *FedNIDS* model encounters challenges in classifying adversarial samples, resulting in an average F1 score of 0.17 across all silos for both toolchains. This difficulty can be attributed to the inherent complexity of identifying adversarial evasion samples within the federated learning global model. However, after the fine-tuning process in Stage 2, *FedNIDS* demonstrates a noteworthy improvement in accuracy against adversarial samples. The average F1 score across all silos increases to 0.92. The fine-tuning mechanism enables the model to adapt and refine its decision boundaries based on the additional adversarial data, resulting in a significant improvement (around 75%) in the F1 score. This process enhances the model's ability to learn more robust features and decision rules, ultimately enabling it to better discriminate between benign and adversarial samples. These findings hold significant implications for real-world deployment scenarios where NIDS frequently faces adversarial attempts. These experiments underscore the effectiveness of our *FedNIDS* methodology in providing an adaptive and robust defense against novel network threats while maintaining data privacy and security.

## 6 Conclusions and Future Works

In this study, we have presented the *FedNIDS*, a novel framework that leverages the power of federated learning and DNNs to enhance the accuracy, robustness, and privacy preservation of NIDS. Our study is the first to address the issue of non-IID data across different clients for packet-based NIDS in a federated learning framework. The framework addresses the challenge of accurate attack detection while respecting the privacy constraints of individual clients. The *FedNIDS* framework comprises two stages: federated DNN pre-training and federated novel attack fine-tuning. Through a comprehensive experimental evaluation, we demonstrated the effectiveness of *FedNIDS* in detecting specific attacks across different silos and its ability to adapt to new and emerging attack patterns. The quick adaptability of *FedNIDS* to changing attack patterns ensures that it remains effective even

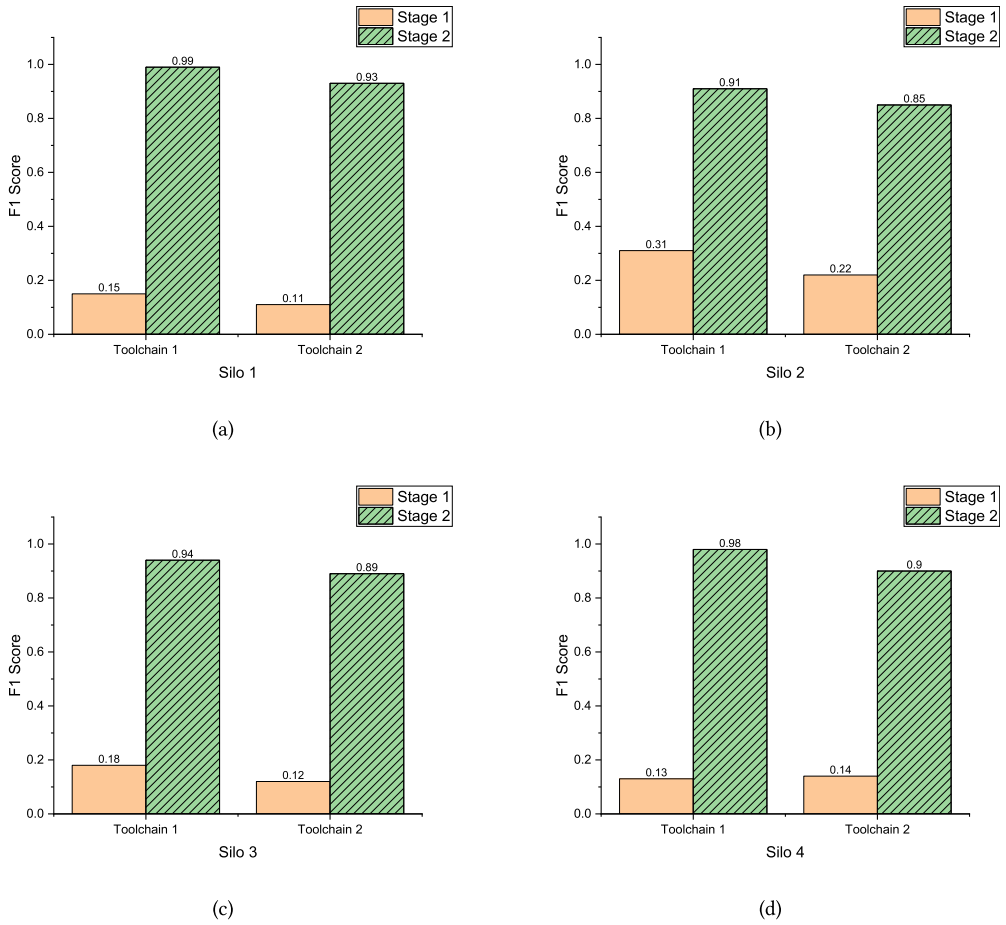


Fig. 4. The performance of FedNIDS against adversarial attacks after fine-tuning.

as new threats emerge. The robustness of our approach against adversarial attacks signifies its suitability for real-world scenarios where adversaries may attempt to evade detection.

Although our study has yielded promising results, below we discuss the assumptions in this study and practical considerations of deploying such a framework:

(1) *Reliable communication*: In our experiments, we assume reliable communication channels between the silos and the central server. In real-world scenarios, factors like latency, bandwidth constraints, or communication costs could pose practical challenges.
(2) *Privacy and security*: Our federated learning approach assumes a strong foundation of privacy and security to ensure that the learned weights transferred out from silos are secured. This is an underlying assumption in our approach and should be appropriately designed in real-world deployments.
(3) *Non-IID data in silos*: While our silo setup simulates a decentralized network, real-world conditions may introduce greater complexity in data distribution. Factors such as diverse hardware, software configurations, and network traffic patterns could intensify non-IID effects. Employing quantitative methods to assess the non-IID nature of the data within silos, such as data augmentation, transfer learning, and domain

adaptation algorithms or advanced statistical analyses, would enhance the robustness of such an approach in a real-world setting.

(4) *Novelty detection and labeling*: Our work assumes the availability of novel attack samples during the fine-tuning stage. However, in the real world, security analysts at the affected silo are tasked with identifying novel attacks that bypass the local model. These newly labeled samples are then used for the fine-tuning stage to further update the global model.

(5) *Dataset availability and proportion of attack samples*: The availability of labeled network data for security purposes is limited, which can pose challenges for training intrusion detection models. Collaboration with industry partners and network security practitioners is crucial to obtain labeled network data from real-world environments. The definition of "malicious" traffic can also be nuanced and context-dependent, and the labeling of network activities as malicious or benign may vary depending on the specific security policies and threat models of an organization. The proportion of malicious packets in our experimental setup obtained from publicly available datasets may not accurately reflect the realistic ratios encountered in live network environments, where malicious traffic often constitutes a small fraction of the overall network traffic. The network traffic volume used in our experiments, measured in gigabytes, may not fully represent the scale and complexity of heavily utilized live networks. To address these concerns, adaptive labeling and context-aware detection techniques should be developed to enhance the practicality of deploying such a system in different organizational contexts. Strategies to handle class imbalance between benign and malicious traffic in the training data should also be explored to improve the model's performance in real-world scenarios.

While our study focuses on packet-based analysis, future research could explore the integration of flow-based and packet-based approaches to further enhance network security analysis, leveraging the strengths of both techniques. One interesting avenue would be to investigate a packet-level graph-based representation. This representation would incorporate flow and application metadata, offering the intrusion detector additional context. Another avenue is examining alternative DNN architectures suitable for resource-constrained devices like IoT devices and sensors, which grapple with constraints in processing power, memory, storage capacity, and unreliable network connectivity. Furthermore, it would be valuable for the cybersecurity research community to investigate the delicate balance between accuracy and efficiency in these scenarios. Another intriguing avenue is exploring the integration of multi-modal data sources to achieve a more comprehensive understanding of network behaviors.

## References

[1] UNB. 2023. Canadian Institute of Cybersecurity (CIC). Retrieved May 01, 2023 from https://www.unb.ca/cic/datasets/ids-2017.html

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318.

[3] Shaashwat Agrawal, Sagnik Sarkar, Ons Aouedi, Gokul Yenduri, Kandaraj Piamrat, Mamoun Alazab, Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, and Thippa Reddy Gadekallu. 2022. Federated learning for intrusion detection system: Concepts, challenges and future directions. *Computer Communications* 195 (2022), 346–361.

[4] Muhammad Ahmad, Qaiser Riaz, Muhammad Zeeshan, Hasan Tahir, Syed Ali Haider, and Muhammad Safeer Khan. 2021. Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set. *EURASIP Journal on Wireless Communications and Networking* 2021, 1 (2021), 1–23.

[5] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. 2016. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications* 60 (2016), 19–31.

[6] Ammar Alazab, Ansam Khraisat, Sarabjot Singh, and Tony Jan. 2023. Enhancing privacy-preserving intrusion detection through federated learning. *Electronics* 12, 16 (2023), 3382.

[7] Ons Aouedi, Kandaraj Piamrat, Guillaume Muller, and Kamal Singh. 2022. FLUIDS: Federated learning with semi-supervised approach for Intrusion Detection System. In *Proceedings of the 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC '22)*. IEEE, 523–524.

[8] Manjula C. Belavagi and Balachandra Muniyal. 2016. Performance evaluation of supervised machine learning algorithms for intrusion detection. *Procedia Computer Science* 89 (2016), 117–123.

[9] Aitor Belenguer, Jose A. Pascual, and Javier Navaridas. 2023. GöwFed: A novel federated network intrusion detection system. *Journal of Network and Computer Applications* 217 (2023), 103653.
[10] Vance W. Berger and YanYan Zhou. 2014. Kolmogorov–Smirnov test: Overview. *Wiley Statsref: Statistics Reference Online*. Wiley Online Library
[11] David A. Bierbrauer, Michael J. De Lucia, Krishna Reddy, Paul Maxwell, and Nathaniel D. Bastian. 2023. Transfer learning for raw network traffic detection. *Expert Systems with Applications* 211 (2023), 118641.
[12] Philippe Biondi. 2010. Scapy documentation (!). Vol. 469 (2010), 155–203.
[13] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 1175–1191.
[14] Jie Cui, Hu Sun, Hong Zhong, Jing Zhang, Lu Wei, Irina Bolodurina, and Debiao He. 2023. Collaborative intrusion detection system for SDVN: A fairness federated deep learning approach. *IEEE Transactions on Parallel and Distributed Systems* 34, 9 (2023), 2512–2528.
[15] Michael J. De Lucia, Paul E. Maxwell, Nathaniel D. Bastian, Ananthram Swami, Brian Jalaian, and Nandi Leslie. 2021. Machine learning raw network traffic detection. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, Vol. 11746. SPIE, 185–194.
[16] Felix Erlacher and Falko Dressler. 2018. FIXIDS: A high-speed signature-based flow intrusion detection system. In *Proceedings of the NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–8.
[17] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security* 28, 1–2 (2009), 18–28.
[18] Jalal Ghadermazi, Ankit Shah, and Nathaniel Bastian. 2023. Towards real-time network intrusion detection with image-based sequential packets representation. TechRxiv Preprint. DOI: https://doi.org/10.36227/techrxiv.23291588.v1
[19] John C. Gower. 1971. A general coefficient of similarity and some of its properties. *Biometrics* 27 (1971), 857–871.
[20] Ke He, Dan Dongseong Kim, and Muhammad Rizwan Asghar. 2023. Adversarial machine learning for network intrusion detection systems: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 25, 1 (2023), 538–566.
[21] Hanan Hindy, David Brosset, Ethan Bayne, Amar Kumar Seeam, Christos Tachtatzis, Robert Atkinson, and Xavier Bellekens. 2020. A taxonomy of network threats and the effect of current datasets on intrusion detection systems. *IEEE Access* 8 (2020), 104650–104675.
[22] Vanlalruata Hnamte and Jamal Hussain. 2023. Dependable intrusion detection system using deep convolutional neural network: A Novel framework and performance evaluation approach. *Telematics and Informatics Reports* 11 (2023), 100077. DOI: https://doi.org/10.1016/j.teler.2023.100077
[23] Soumyadeep Hore, Jalal Ghadermazi, Diwas Paudel, Ankit Shah, Tapas K. Das, and Nathaniel D. Bastian. 2023. Deep PackGen: A deep reinforcement learning framework for adversarial network packet generation. arXiv:2305.11039. Retrieved from https://doi.org/10.48550/arXiv.2305.11039
[24] Soumyadeep Hore, Quoc Nguyen, Yulun Xu, Ankit Shah, Nathaniel Bastian, and Trung Le. 2023. Empirical evaluation of autoencoder models for anomaly detection in packet-based NIDS. In *Proceedings of the 2023 6th IEEE Conference on Dependable and Secure Computing*. IEEE, 1–8.
[25] Meryem Janati Idrissi, Hamza Alami, Abdelkader El Mahdaouy, Abdellah El Mekki, Soufiane Oualil, Zakaria Yartaoui, and Ismail Berrada. 2023. Fed-ANIDS: Federated learning for anomaly-based network intrusion detection systems. *Expert Systems with Applications* 234 (2023), 121000.
[26] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi. Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.
[27] Min-Joo Kang and Je-Won Kang. 2016. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one* 11, 6 (2016), e0155781.
[28] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning*. Hal Daumé III and Aarti Singh (Eds.), Proceedings of Machine Learning Research, Vol. 119, PMLR, 5132–5143. DOI: https://proceedings.mlr.press/v119/karimireddy20a.html
[29] Christopher Krügel, Thomas Toth, and Engin Kirda. 2002. Service specific anomaly detection for network intrusion detection. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, 201–208.
[30] Vinod Kumar and Om Prakash Sangwan. 2012. Signature based intrusion detection system using SNORT. *International Journal of Computer Applications & Information Technology* 1, 3 (2012), 35–41.

[31] Beibei Li, Yuhao Wu, Jiarui Song, Rongxing Lu, Tao Li, and Liang Zhao. 2020. DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems. *IEEE Transactions on Industrial Informatics* 17, 8 (2020), 5615–5624.

[32] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (2021), 3347–3366.

[33] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* 2 (2020), 429–450.

[34] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. 2020. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing* 24, 3 (2020), 1999–2012.

[35] Patrick E. McKnight and Julius Najab. 2010. Mann-Whitney U Test. *The Corsini Encyclopedia of Psychology*, 1–1.

[36] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[37] Mohamed Amine Merzouk, Frédéric Cuppens, Nora Boulahia-Cuppens, and Reda Yaich. 2023. Parameterizing poisoning attacks in federated learning-based intrusion detection. In *Proceedings of the 18th International Conference on Availability, Reliability and* Security, 1–8.

[38] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. 2002. Intrusion detection using neural networks and support vector machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN '02) (Cat. No. 02CH37290)*, Vol. 2, IEEE, 1702–1707.

[39] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N. Asokan, and Ahmad-Reza Sadeghi. 2019. DÏoT: A federated self-learning anomaly detection system for IoT. In *Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS '19)*. IEEE, 756–767.

[40] Guansong Pang, Chunhua Shen, and Anton van den Hengel. 2019. Deep anomaly detection with deviation networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 353–362.

[41] Sawsan Abdul Rahman, Hanine Tout, Chamseddine Talhi, and Azzam Mourad. 2020. Internet of things intrusion detection: Centralized, on-device, or federated learning? *IEEE Network* 34, 6 (2020), 310–317.

[42] Md Mamunur Rashid, Shahriar Usman Khan, Fariha Eusufzai, Md Azharuddin Redwan, Saifur Rahman Sabuj, and Mahmoud Elsharief. 2023. A federated learning-based approach for improving intrusion detection in industrial internet of things networks. *Network* 3, 1 (2023), 158–179.

[43] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H. Brendan McMahan. 2020. Adaptive federated optimization. arXiv:2003.00295. Retrieved from https://doi.org/10.48550/arXiv.2003.00295

[44] Pedro Ruzafa-Alcázar, Pablo Fernández-Saura, Enrique Mármol-Campos, Aurora González-Vidal, José L. Hernández-Ramos, Jorge Bernal-Bernabe, and Antonio F. Skarmeta. 2021. Intrusion detection based on privacy-preserving federated learning for the industrial IoT. *IEEE Transactions on Industrial Informatics* 19, 2 (2021), 1145–1154.

[45] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2019. A detailed analysis of the cicids2017 data set. In *Information Systems Security and Privacy: 4th International Conference (ICISSP '18)*. Springer, 172–188.

[46] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization (*ICISSp '18)* 1 (2018), 108–116.

[47] Dug Song and Contributors. 2009. dpkt 1.9.2 documentation. Retrieved February 16, 2023 from https://dpkt.readthedocs.io/en/latest/

[48] K. Muthamil Sudar and P. Deepalakshmi. 2022. Flow-based detection and mitigation of low-rate DDOS attack in sdn environment using machine learning techniques. In *IoT and Analytics for Sensor Networks: Proceedings of ICWSNUCA 2021*. Springer, 193–205.

[49] Pu Tian, Zheyi Chen, Wei Yu, and Weixian Liao. 2021. Towards asynchronous federated learning based threat detection: A DC-Adam approach. *Computers & Security* 108 (2021), 102344.

[50] Muhammad Fahad Umer, Muhammad Sher, and Yaxin Bi. 2017. Flow-based intrusion detection: Techniques and challenges. *Computers & Security* 70 (2017), 238–254.

[51] Nguyen Thanh Van, Tran Ngoc Thinh, and Le Thanh Sach. 2017. An anomaly-based network intrusion detection system using deep learning. In *Proceedings of the 2017 International Conference on System Science and Engineering (ICSSE '17)*. IEEE, 210–214.

[52] Miel Verkerken, Laurens D'hooge, Tim Wauters, Bruno Volckaert, and Filip De Turck. 2022. Towards model generalization for intrusion detection: Unsupervised machine learning techniques. *Journal of Network and Systems Management* 30 (2022), 1–25.

[53] Bo Wang, Yang Su, Mingshu Zhang, and Junke Nie. 2020. A deep hierarchical network for packet-level malicious traffic detection. *IEEE Access* 8 (2020), 201728–201740.

[54] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2020. Federated learning with matched averaging. arXiv:2002.06440. Retrieved from https://doi.org/10.48550/arXiv.2002.06440

[55] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in Neural Information Processing Systems* 33 (2020), 7611–7623.

[56] Mikal R. Willeke, David A. Bierbrauer, and Nathaniel D. Bastian. 2023. Data-efficient, federated learning for raw network traffic detection. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications V*, Vol. 12538, SPIE, 247–262.

[57] Kai Yang, Yuanming Shi, Yong Zhou, Zhanpeng Yang, Liqun Fu, and Wei Chen. 2020. Federated machine learning for intelligent IoT via reconfigurable intelligent surface. *IEEE Network* 34, 5 (2020), 16–22.
[58] Run Yang, Hui He, Yulong Wang, Yue Qu, and Weizhe Zhang. 2023. Dependable federated learning for IoT intrusion detection against poisoning attacks. *Computers & Security* 132 (2023), 103381.
[59] Binhang Yuan, Song Ge, and Wenhui Xing. 2020. A federated learning framework for healthcare IoT devices. arXiv:2005.05083. Retrieved from https://doi.org/10.48550/arXiv.2005.05083
[60] Si-si Zhang, Jian-wei Liu, and Xin Zuo. 2021. Adaptive online incremental learning for evolving data streams. *Applied Soft Computing* 105 (2021), 107255.

Received 10 November 2023; revised 26 July 2024; accepted 28 August 2024