COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS DE LA LOIRE – IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Sciences et technologies de l'information et de la communication*

Par
Léo LAVAUR

Améliorer la détection d'intrusions dans les systèmes répartis grâce à l'apprentissage fédéré

Thèse présentée et soutenue à Rennes, le 7 octobre 2024
Unité de recherche : IRISA (UMR 6074), SOTERN

Rapporteurs avant soutenance :

Anne-Marie KERMARREC	Professeure à l'Université Polytechnique Fédérales de Lausanne (EPFL)
Éric TOTEL	Professeur à Télécom SudParis

Composition du Jury :

Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer quelle est conforme et devra être répercutée sur la couverture de thèse

Président :	*À compléter après la soutenance.*	
Examinateurs :	Sonia BEN MOKHTAR	Directrice de Recherche au CNRS
	Pierre-François GIMENEZ	Maître de Conférence à CentraleSupelec
	Vincent NICOMETTE	Professeur à INSA Toulouse
	Fabien AUTREL	Ingénieur de Recherche à IMT Atlantique
	Marc-Oliver PAHL	Directeur d'Études à IMT Atlantique
Dir. de thèse :	Yann BUSNEL	Professeur à IMT Nord Europe

Invité(s) :

Prénom NOM	Fonction et établissement d'exercice

Abstracts

Résumé

La collaboration entre les différents acteurs de la cybersécurité est essentielle pour lutter contre des attaques de plus en plus sophistiquées et nombreuses. Pourtant, les organisations sont souvent réticentes à partager leurs données, par peur de compromettre leur confidentialité et leur avantage concurrentiel, et ce même si cela pourrait d'améliorer leurs modèles de détection d'intrusions. L'apprentissage fédéré est un paradigme récent en apprentissage automatique qui permet à des clients répartis d'entraîner un modèle commun sans partager leurs données. Ces propriétés de collaboration et de confidentialité en font un candidat idéal pour des applications sensibles comme la détection d'intrusions. Si un certain nombre d'applications ont montré qu'il est, en effet, possible d'entraîner un modèle unique sur des données réparties de détection d'intrusions, peu se sont intéressées à l'aspect collaboratif de ce paradigme. Dans ce manuscrit, nous étudions l'utilisation de l'apprentissage fédéré pour construire des systèmes collaboratifs de détection d'intrusions. En particulier, nous explorons (i) l'impact de la qualité des données dans des contextes hétérogènes, (ii) l'exposition à certains types d'attaques par empoisonnement, et (iii) des outils et des méthodologies pour améliorer l'évaluation de ce type d'algorithmes.

Abstract

Collaboration between different cybersecurity actors is essential to fight against increasingly sophisticated and numerous attacks. However, stakeholders are often reluctant to share their data, fearing confidentiality and privacy issues and the loss of their competitive advantage, although it would improve their intrusion detection models. Federated learning is a recent paradigm in machine learning that allows distributed clients to train a common model without sharing their data. These properties of collaboration and confidentiality make it an ideal candidate for sensitive applications such as intrusion detection. While several applications have shown that it is indeed possible to train a single model on distributed intrusion detection data, few have focused on the collaborative aspect of this paradigm. In this manuscript, we study the use of federated learning to build collaborative intrusion detection systems. In particular, we explore (i) the impact of data quality in heterogeneous contexts, (ii) the exposure to certain types of poisoning attacks, and (iii) tools and methodologies to improve the evaluation of these types of algorithms.

ACKNOWLEDGEMENTS

Table of Contents

Chapter 1

Introduction

Contents

1.1 Context and Motivation

Modern information security is made difficult by the scale, complexity, and heterogeneity of information systems. Because security by design in these conditions is a considerable challenge, security agencies also recommend complementary measures. For instance, the NIST Cybersecurity Framework [Nat24] suggests a five-stage lifecycle for managing risks in information systems: identify, protect, detect, respond, and recover.

The *detection* and *response* stages can significantly benefit from the recent advances in Artificial Intelligence (AI) and Machine Learning (ML), enabling the analysis of more complex behaviors. Yet, because organizations usually face similar threats, including large-scale campaigns such as Mirai in 2016 or NotPetya in 2017, they would greatly benefit from sharing insights on the intrusions they have encountered, or any knowledge that might help others to identify the incident before the damages are too important. Collaboration is further encouraged by regulation, for instance with the NIS [16] and NIS2 [22] European directives. Sharing data is made even more important for training ML and Deep Learning (DL) models, which require large amounts of data to be effective. Yet, stakeholders are often reluctant to involve their organization in data-sharing practices, fearing confidentiality and privacy breaches, reputation loss, or regulation non-compliance.

Federated Learning (FL) [McM+17] has emerged as a promising paradigm for collaborative ML, enabling model training across distributed data sources while preserving privacy. Deployed in intrusion detection contexts, FL can help organizations to virtually extend the size of their training sets, thus producing more accurate models. This architecture could also be used to disseminate information about esoteric attacks or devices behavior owned locally, that would benefit to other organizations. FL also promises to solve other drawbacks of ML-based Intrusion Detection Systems (IDSs), such as the need

for continuous retraining, the lack of adaptability to new threats, or the risk of local biases due to a lack of heterogeneity in the training set.

Consequently, applying FL to IDS seems like a promising approach to collaboratively improve the local detection of cyber threats. This is supported by the amount of recent literature on the topic, which has grown exponentially since 2018 [Ism+24; Lav+22b]. Yet, novel challenges arise in this context, such as how to handle the heterogeneity of data sources or how to deal with untrusted participants. But more importantly, *what makes applying FL to IDS different from other applications? And is FL even a suitable framework for collaborative IDS?*

This dissertation aims to investigate the potential of Federated Learning as a collaborative framework for Intrusion Detection System, which we will refer to as Federated Intrusion Detection System (FIDS). The remaining of this manuscript will discuss the state of the art in FL and IDS, some of the challenges that arise in this context, and the potential solutions to address them.

1.1.1 Use case boundaries

While applying FL to IDS can already be considered as a restricted scope, the IDS literature contains a wide variety of use cases, each coming with its own set of specificities and constraints. For instance, IDSs can be deployed at the network level, the host level, or the application level. Likewise, objectives and constraints may vary depending on the context and the type of devices involved: Internet of Things (IoT), Industrial Control System (ICS), or traditional information systems. Among the most common combinations, Network-based Intrusion Detection System (NIDS) on Information Technology (IT) network data stands out, notably in terms of implemented algorithms and available datasets. This is particularly important for evaluation purposes, as it makes it easier to compare the performance of different approaches.

Additionally, this use case provides a realistic application for FIDSs, where the actors are organizations that own or oversee an information system, and that are interested in improving their local detection. This is typically referred to as Collaborative Intrusion Detection System (CIDS). For instance, Security Operations Centers (SOCs) monitor the network traffic of their customers for security purposes, and cannot afford to share this data with other organizations. Two SOCs could, however, share insights on the threats they have encountered, or the behaviors they have observed, without sharing the raw data. Existing structures, such as Information Sharing and Analysis Centers (ISACs) or inter-SOCs could benefit from such a framework, as they already have a trust relationship with their members.

Consequently, this dissertation will focus on the use case of building collaborative NIDSs by leveraging FL on IT network data. Note however, that the results presented in

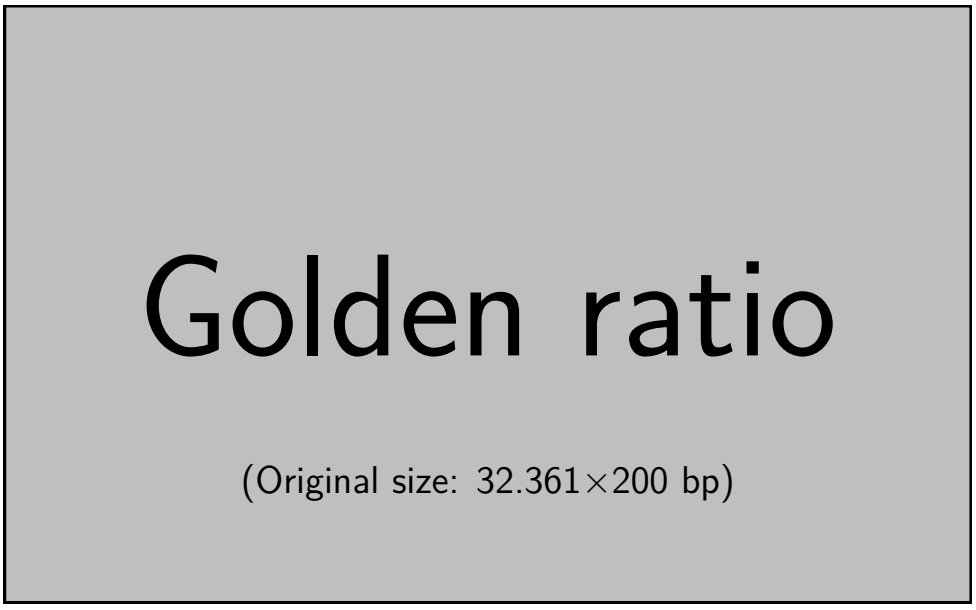


Figure 1.1 – Illustration of FL in a CIDS use case.

this manuscript could theoretically be extended to other applications. Figure 1.1 illustrates our use case.

1.2 Research Objectives

Overall, this work aims to answer the following: *Can FL serve as a trustable knowledge-sharing framework for collaboratively improving intrusion detection mechanisms?* Based on the context and motivation laid out in the previous section, we formulize the general objectives of this dissertation as a set of research questions. The questions stated hereafter are intended to be completed and extended in the following chapters, some of which introduce their own research questions.

Specifically, we globally focus on the following research questions:

RQ1. *What makes applying FL to IDSs specific?*

RQ2. *Can FL be used to federate IDSs across heterogeneous data sources?*

RQ3. *How does FL handle malicious contributions in a federated IDS?*

RQ4. *How can we address the main challenges in applying FL to IDS?*

1.3 Contributions

We summarize the contributions of this dissertation as follows:

1. The first Systematic Literature Review (SLR) in literature that study the application of FL to IDS. We propose a reference architecture and a taxonomy for structur-

ing the domain, supported by quantitative and qualitative analyses of the existing literature.

2. An illustrative study highlighting the challenges of heterogeneity and malicious contributions in FIDS.
3. An extensible evaluation framework for FIDSs called Eiffel, leveraging popular open source libraries like Flower [Beu+20] and Hydra [Yad19], and a set of malicious clients simulators.
4. A systematic analysis of the impact of label-flipping attacks on an FL-based collaborative IDS, leveraging the aforementioned evaluation framework.
5. A pioneer FL architecture for collaborative IDS that handles malicious contributions in heterogeneous environments, leveraging a cross-evaluation mechanism and a reputation system.
6. A methodology allowing to generate network topologies with heterogeneity constraints, and laying down the foundations toward a more realistic evaluation of FIDS and distributed networking telemetry experiments in general.

1.4 Outline

Apart from the introduction and conclusion chapters, the manuscript is organized in two parts: we define FIDSs and identify their limitations in Part I, before quantifying their limitations and providing solutions to address them in Part II.

Part I: The first part delves into the application of FL to IDS. After layout out the necessary background in Chapter 2, we present the state of the art in FIDS in Chapter 3. This chapter notably presents the results of our SLR on the topic, and focus on the related challenges and research opportunities. Chapter 4 then closes this first part by highlighting the main challenges in FIDS using toy examples.

Part II: The second part presents our contributions to addressing the limitations of FIDS. Finally, Chapter 5 introduces our evaluation framework, and systematically analyses the impact of label-flipping attacks on FIDS. Chapter 6 then presents a pioneer FL architecture for FIDS that ensures the quality of incoming contributions in heterogeneous environments, with applications to the detections of malicious behaviors. Finally, Chapter 7 introduces a practical method to generate network topologies based on the composition of sub-topologies, and lays down the foundations for further studies on distributed networking analyses.

1.5 Publications

Journal articles

[Léo+22b] **Léo Lavaur**, Marc-Oliver Pahl, Yann Busnel, and Fabien Autrel, « The Evolution of Federated Learning-based Intrusion Detection and Mitigation: A Survey », *in*: *IEEE Transactions on Network and Service Management*, Special Issue on Network Security Management (June 2022).

International conference papers

[Léo+24] **Léo Lavaur**, Pierre-Marie Lechevalier, Yann Busnel, Romaric Ludinard, Géraldine Texier, and Marc-Oliver Pahl, « RADAR: Model Quality Assessment for Reputation-aware Collaborative Federated Learning », *in*: *Proceedings of the 43rd International Symposium on Reliable Distributed Systems (SRDS)*, Charlotte, NC, USA, Sept. 2024.

[LBA24a] **Léo Lavaur**, Yann Busnel, and Fabien Autrel, « Systematic Analysis of Label-flipping Attacks against Federated Learning in Collaborative Intrusion Detection Systems », *in*: *Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES), Workshop on Behavioral Authentication for System Security (BASS)*, Vienna, Austria, Aug. 2024.

[BL24] Yann Busnel and **Léo Lavaur**, « Tutorial: Federated Learning × Security for Network Monitoring », *in*: *Proceedings of the 44th International Conference on Distributed Computing Systems (ICDCS)*, Jersey City, NJ, USA, July 2024.

[LBA24b] **Léo Lavaur**, Yann Busnel, and Fabien Autrel, « Demo: Highlighting the Limits of Federated Learning in Intrusion Detection », *in*: *Proceedings of the 44th International Conference on Distributed Computing Systems (ICDCS)*, Jersey City, NJ, USA, July 2024.

National conference papers

[Léo+23] **Léo Lavaur**, Pierre-Marie Lechevalier, Yann Busnel, Marc-Oliver Pahl, and Fabien Autrel, « Metrics and Strategies for Adversarial Mitigation in Federated Learning-based Intrusion Detection », *in*: *Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information (RESSI)*, Neuvy-sur-Barangeon, France, May 2023.

[Léo+22a] **Léo Lavaur**, Benjamin Coste, Marc-Oliver Pahl, Yann Busnel, and Fabien Autrel, « Federated Learning as Enabler for Collaborative Security between Not Fully-Trusting Distributed Parties », *in*: *Proceedings of the 29th Computer & Electronics Security Application Rendezvous (C&ESAR)*, Rennes, France, Oct. 2022.

[Léo+21] **Léo Lavaur**, Marc-Oliver Pahl, Yann Busnel, and Fabien Autrel, « Federated Security Approaches for IT and OT », *in*: *Journée thématique du GT sur la Sécurité des Systèmes, Logiciels et Réseaux (GT-SSLR)*, May 2021.

Tutorials

[BL23a] Yann Busnel and **Léo Lavaur**, « Federated Learning × Security for Network Management », 15th International Conference on Network of the Future (NoF), Izmir, Turkey, Sept. 2023.

[BL23b] Yann Busnel and **Léo Lavaur**, « L'interêt de l'apprentissage fédéré dans le cadre de la détection d'incidents sur les réseaux et/ou systèmes à grande échelle », Ecole de Printemps Recherche de l'EUR CyberSchool, Rennes, France, Apr. 2023.

Part I

Federated Learning to build CIDSs

Chapter 2

Background and Preliminaries

Contents

2.1 Introduction

This chapter provides the necessary background and preliminaries to navigate the rest of the manuscript. We start by laying out the foundations of Machine Learning (ML) for intrusion detection in Section 2.2, followed by the implications of scaling up to Collaborative Intrusion Detection Systems (CIDSs) in Section 2.3, enumerating along the way the challenges that motivate the use of Federated Learning (FL). We then introduce the fundamentals of FL in Section 2.4, focusing on the FedAvg algorithm and the notations used throughout the thesis. Finally, we discuss the threats against FL in Section 2.4.4, with a particular focus on data poisoning attacks.

2.2 Intrusion Detection

Organizations long relied on signature-based Intrusion Detection Systems (IDSs) to detect intrusions. These systems leverage a database of known attack patterns (*i.e.*, *signatures*) to identify malicious activities. Listing 2.1 displays an example of a signature for detecting the Heartbleed vulnerability using Suricata [The], a popular open-source IDS. This signature relies on dedicated code inside Suricata's engine that required extensive human intervention to develop. It is consequently specific to Suricata and remains difficult to interpret and adapt. Such limitations motivated the study of ML for automatically extracting patterns from data, enabling the development of more flexible and adaptive IDSs.

IDSs can be broadly classified into two categories: *misuse detection* and *anomaly detection*. Misuse detection refers to the identification of known attack patterns. Signature-

1. https://github.com/OISF/suricata/blob/master/rules/tls-events.rules

```
alert tls any any -> any any (msg:"SURICATA TLS overflow heartbeat
    encountered, possible exploit attempt (heartbleed)"; flow:established;
    app-layer-event:tls.overflow_heartbeat_message; flowint:tls.anomaly.
    count,+,1; classtype:protocol-command-decode; reference:cve,2014-0160;
    sid:2230012; rev:1;)
```

Listing 2.1 – Example of a Suricata signature for detecting the Heartbleed vulnerability [1].

based IDSs fall into this category. On the other hand, anomaly detection compares a normal profile (trained on nominal traffic) with observed events to determine if they are malicious [Gar+09]. This approach is *de facto* more efficient for detecting novel attacks, but it is also more prone to false positives.

An analogy can be drawn between this classification and the two main paradigms of ML: supervised and unsupervised learning. In supervised learning, the model is trained on labeled data, where each sample is associated with a label. Labels can be classes (binary or multi-class classification) or continuous values (regression). Either way, the model's objective is to predict the label of unseen samples. In unsupervised learning, no labels are provided. The model's goal is to find patterns in the data, such as clusters or outliers. In anomaly detection, the model is trained on normal data only, and its objective is to detect deviations from this normal profile.

Multiple ML algorithms have been applied to intrusion detection, including Support Vector Machines (SVMs), Random Forests (RFs), and Artificial Neural Networks (ANNs). However, the rise of Deep Learning (DL) has led to a significant improvement in the performance of IDSs. DL is a subfield of ML that focuses on learning representations of data through the use of neural networks. In the following sections, we introduce the basics of DL for intrusion detection, review the existing paradigms, and discuss the metrics and datasets used to evaluate these systems.

2.2.1 Deep Learning for Intrusion Detection

DL present several advantages over traditional ML algorithms. Most notably, they automatically learn features from the data, reducing the need for manual feature engineering. This is particularly useful in the context of intrusion detection, where the features are often complex, interdependent, and of unequal relevance. The training data can range from network traffic to system logs depending on the type of mechanism used: network-based, host-based, or hybrid. Yet, Network-based Intrusion Detection Systems (NIDSs) greatly outnumber other approaches in the literature, due the availability of network traffic datasets and the ease of deployment.

Most of the research on NIDS use a representation known as *unidirectional network flows* or *netflows*, where a flow is defined as a sequence of packets sharing the same source

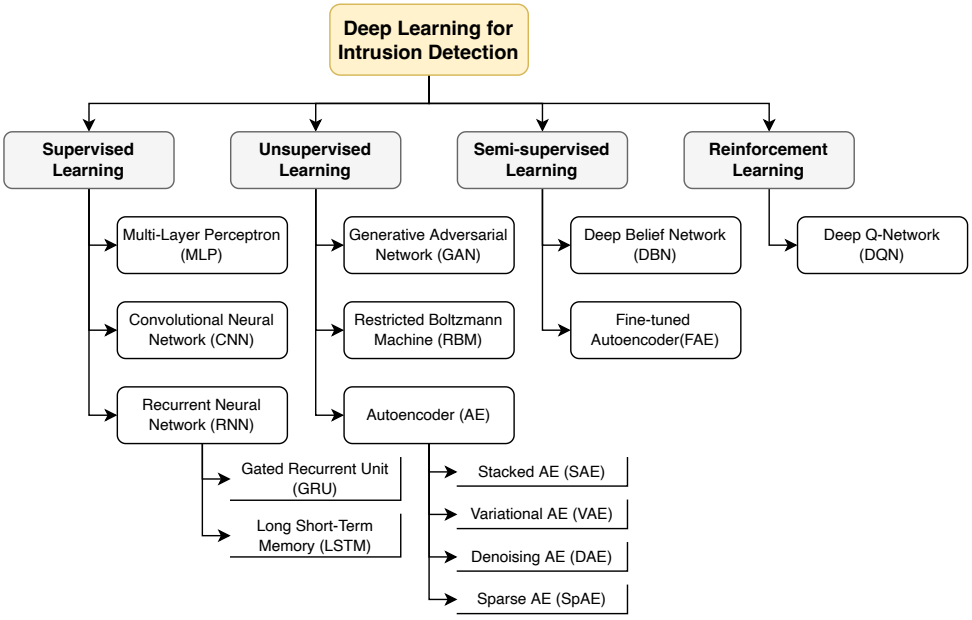


Figure 2.1 – Taxonomy of the main DL paradigms.

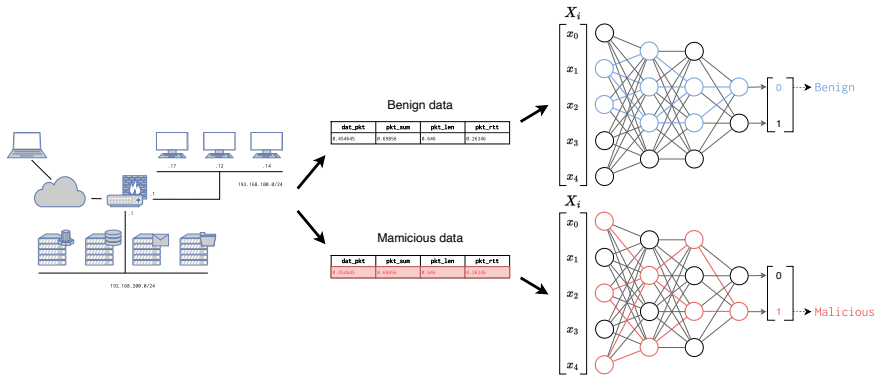
and destination addresses, ports, and protocol. Various features can be extracted from these flows, such as the number of packets, bytes, and the duration of the flow. More details on the features used in NIDS can be found in Section 2.2.2. More generally, the dataset D of size n is represented as a set of variables $X_i = \langle x_1, x_2, \ldots, x_m \rangle, i \in [\![1, n]\!]$, where x_j corresponds to the j-th feature, and m to the number of features.

Main DL Paradigms

Because of their layered architecture, Deep Neural Networks (DNNs) can adopt different forms depending on the type of input data and the task at hand. Figure 2.1 presents the major families of DL algorithms: supervised, unsupervised, and semi-supervised learning, and finally reinforcement learning. While works exist on the application of reinforcement learning to intrusion detection [He+24], they remain rare in the literature. Consequently, we focus on supervised and unsupervised learning in this thesis. This section provides an overview of these paradigms, and define for each the learning problem in the context of intrusion detection.

Supervised Learning Supervised learning is the most common approach in ML, and refers to the training of a model on labeled data. In the context of IDSs, practitioners usually seek to classify network flows into two classes (*benign* and *malicious*), which is a binary classification task. Consequently, the dataset D of size n associates each sample X_i with a label $y_i \in \{0, 1\}$. The model is trained to predict the label $\hat{y}$ of unseen samples. To do so, we generally use a Stochastic Gradient Descent (SGD)-based optimizer to minimize

(a) Training phase.



(b) Prediction phase.

Figure 2.2 – Workflow of a Multilayer Perceptron (MLP) for intrusion detection.

a loss function

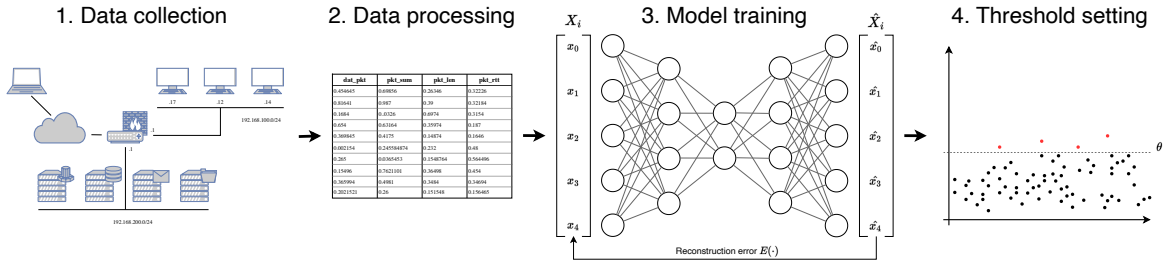
$$\mathcal{L}(w, X_i, y_i), i \in [\![1, n]\!], \tag{2.1}$$

where w represent the model's parameters. After computing the gradients $\nabla\mathcal{L}(w, X_i, y_i)$, they can update their model as

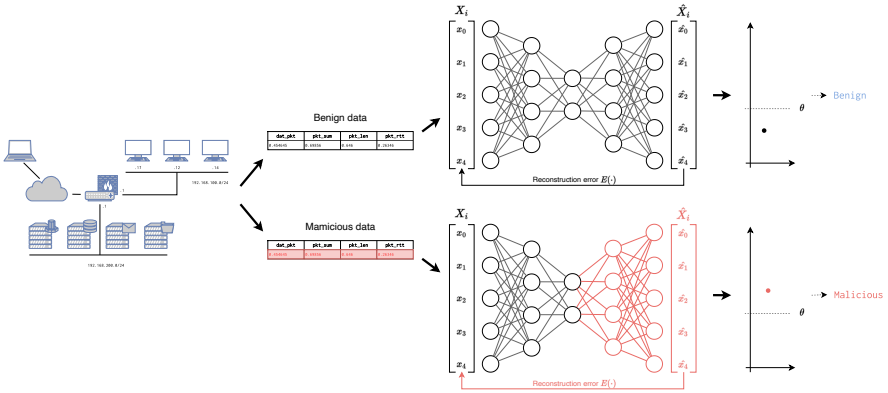
$$w^{t+1} \leftarrow w^t - \eta\nabla\mathcal{L}(w, X_i, y_i), \tag{2.2}$$

where η is the learning rate, w^t the model's parameters at iteration t, and w^{t+1} the new parameters resulting from the update. The last layer usually uses `softmax` or `sigmoid` activation functions to output a probability of being in a class (normal or abnormal). In the case of multi-class classification, the label is one-hot encoded [2] into a vector Y of size c (the number of classes), and the `softmax` function is used. Depending on the available features and the learning objective, various architectures can be used, such as Convolutional Neural Networks (CNNs) for high-dimensional data, or Recurrent Neural Networks (RNNs) for sequential data. Multilayer Perceptrons (MLPs) are the simplest and most common architecture used in IDS, and the one that we focus on in this thesis, although most concepts can be extended to other architectures.

2. One-hot encoding is a binary representation of categorical variables, where each category is mapped to a binary vector. It is typically used in ML to represent categorical data, such as the protocol type in netflows.



(a) Training phase.



(b) Prediction phase.

Figure 2.3 – Workflow of a Stacked Autoencoder (SAE) for intrusion detection.

One of the main challenges in supervised learning is the availability of labeled data and its quality. In the context of IDS, obtaining enough labeled data is particularly challenging, as labeling requires expert knowledge and is time-consuming. Moreover, the class distribution is often unbalanced, with benign traffic being much more frequent than anomalies in the testing set [CBK09]. This issue is aggravated in siloed configurations, *i.e.*, in which models can only be trained on locally-collected data. This can lead to models that are skewed by the unbalanced class distribution [Cam+22].

Challenge 1. Locally collected data is often unbalanced, leading to representation biases and overall lower performance.

Unsupervised Learning To circumvent the need for labeled data, unsupervised learning can be used. Unsupervised ML algorithms are typically used for clustering or outlier detection. The DL variants are rather used for feature extraction and dimensionality reduction, or anomaly detection. To detect anomalies, Autoencoders (AEs) can be trained on normal data only, and then used to see whether the reconstruction error of a new sample is above a certain threshold. This builds on the assumption that (i) benign traffic is much more frequent that anomalies in the testing set [CBK09]; and (ii) abnormal packets are statistically different from normal ones. In this scenario, the training dataset D is composed of benign (*i.e.*, normal) samples only and no associated label. Given $\mathcal{X} = \langle X_i | i \in [\![1, n]\!] \rangle$, the model is trained to minimize the reconstruction error $\mathcal{L}(\mathcal{X}, \hat{\mathcal{X}})$,

where $\hat{\mathcal{X}}$ is the output of the AE. A typical error function for this task is the Mean Squared Error (MSE), expressed as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \hat{X}_i\right)^2. \tag{2.3}$$

Then the model can be updated using the same process described in Equation (2.2). Different architectures of AEs can be used, such as Stacked Autoencoders (SAEs) to improve the quality of the extracted features, or Denoising Autoencoders (DAEs) to improve the robustness of the model [GG20]. To detect anomalies, the reconstruction error of a new sample is compared to a threshold θ defined during the training phase on validation data. A high reconstruction error indicates that the considered samples is too *far* from the training data, and can indicate an anomaly. The performance of the combination of the AE and the threshold can then be evaluated using a labelled test set.

While unsupervised learning is particularly useful for detecting novel attacks, it is also more prone to misclassification. Local data in the real-world is likely to be collected on devices with little variance, *e.g.* same brand, same protocols, or use cases. This can lead to a normal profile that is too specific to the local environment, and thus would raise alerts as soon as a change occurs [LL19].

Challenge 2. Local data is specific to the environment, increasing the risk of false positives when changes occur.

Semi-supervised Learning Semi-supervised learning is a hybrid approach where only a small part of the training data is labeled. This approach is particularly useful in the context of IDS, where labeled data is scarce. A common strategy is to train an AE on the full dataset to learn the optimal representation of the data, and use the encoder (see Figure 2.3) part with a classifier to predict the label of the samples [APB20]. Other known model architectures for semi-supervised learning include Deep Belief Networks (DBNs), where multiple layers of Restricted Boltzmann Machines (RBMs) are stacked to form a deep network that extracts features from the data. The model is then fine-tuned using the labeled data for classification purposes.

2.2.2 Datasets

Datasets are essential in intrusion detection, as they allow researchers to evaluate and compare their solutions. This is even more critical when leveraging ML and DL techniques, as the performance of these models is highly dependent on the quality and quantity of the training data. Until the mid-2010s, the most common dataset used for intrusion detection was the KDD'99 dataset [Sig99], built for the KDD Cup 1999 competition using the DARPA 1998 dataset. Tavallaee *et al.* [Tav+09] published an updated version

Table 2.1 – Most common feature-based datasets for NIDSs.

Dataset	Year	Use case	Feature extraction	Features	Records (train/test) [a]	Attack classes	Reference
KDD Cup 99	1999	Military IT Network	Bro-IDS	41	4,898,431/311,029	4	[Sig99]
NSL-KDD	2009	Military IT Network	See KDD'99	41	125,973/22,544	4	[Tav+09]
UNSW-NB15	2015	Company IT Network	Bro, Argus, & Custom	49	2,540,044	10	[MS15]
CIDDS-001	2017	Small Business	NetFlow v9	10	31,959,175	5	[Rin+17a]
CIDDS-002	2017	Small Business	NetFlow v9	10	16,161,183	5	[Rin+17b]
CICIDS2017	2017	Company IT Network	CICFlowMeter	80	2,830,743	9	[SHG18]
CICIDS2018	2018	Large-scale IT Network	CICFlowMeter	80	8,284,254	7	[SHG18]
Bot-IoT	2019	Botnets and IoT	Argus & Custom	14	72,000,000+	4	[Kor+19]
ToN_IoT	2021	Cross-layer Infrastructure	Zeek & Custom	44	461,043	9	[Mou21]
Edge-IIoTset	2022	Cross-layer Infrastructure	Zeek & TShark	61	181,156/30,440	15	[Fer+22]

a. Some datasets do not have a recommended train/test split. In such cases, only the total number of records is provided.

of the dataset, called NSL-KDD, which removes duplicates and corrects some errors in the original dataset. However, NSL-KDD is still based on the original DARPA 1998 dataset, and is considered outdated by today's standards.

Since 2015 with the publication of the UNSW-NB15 dataset [MS15], new datasets have been developed to address the limitations of the KDD'99 and NSL-KDD datasets, such as the lack of realism [3] of the generated traffic, the lack of attack diversity, and the scale of the experiments. Table 2.1 presents the most common feature-based datasets for NIDSs, along with their characteristics. Two teams have been particularly active in this area: the Intelligent Security Group (ISG) [Kor+19; Mou21; MS15] at the University of New South Wales, Australia, and the Canadian Institute for Cybersecurity (CIC) [SHG18] at the University of New Brunswick, Canada. They brought the most used datasets in the field in recent years, UNWS-NB15 and CICIDS2017, respectively.

Provided features Because most of the datasets presented in Table 2.1 are made to train and evaluate MLs models, they rely on a set of features extracted from the network traffic. Some also include the original network captures (PCAPs) for further analysis, or complementary system logs for correlation purposes. Two non-exclusive approaches can be used to produce these features: feature extraction and feature selection.

Feature extraction: It refers to the computation of numerical characteristics after the data collection; *e.g.*, Inter-Arrival Time (IAT) or number of packets per device in the context of traffic monitoring. Most modern dataset use existing IDSs to extract these features, such as Zeek [4] or Argus [5]. The resulting data are network flows, aggregating the information of multiple packets into a single record.

Feature selection: It relates to the selection of the relevant features for a given task.

3. Only in regard to modern networks. Indeed, the DARPA 1998 dataset simulates multiple workstations in a military environment, using the US Air Force Research Laboratory's testbed. The technologies deployed where representative of the state of the art at the time.
4. Formerly known as Bro, available at: https://www.zeek.org/
5. Available at: https://openargus.org/argus-ids

This is particularly useful in the context of ML, where irrelevant or redundant features can degrade the performance of the model. For instance, Edge-IIoTset [Fer+22] contains 61 features, selected from a pool of 1176 using feature correlation.

The choice of features is critical for the performance of the model, although DL models make this process less relevant due to their ability to filter out irrelevant features. Yet, because each dataset comes with its own set of features, it is difficult to compare the performance of models across datasets. Recently, Sarhan, Layeghy, and Portmann [SLP22] proposed a standardized feature set for intrusion detection based on NetFlow V9 [Cla04] format. They used nProbe [6] to convert four known IDS datasets to this format: *i.e.*, UNSW-NB15 [MS15], Bot-IoT [Kor+19], ToN_IoT [Mou21], and CSE-CIC-IDS2018 [SHG18]. The converted datasets (that the author call NF-V2) contain 43 features extracted from flow characteristics, such as duration or packet length, and some others that are protocols-specific. The uniform feature set across datasets allows the evaluation of ML models across independently generated datasets.

Use cases Until 2017 included, most datasets aim at simulating a *typical* network environment, such as deployed in an organization. This is the case for KDD'99, NSL-KDD, UNSW-NB15, and CIDDS 1 and 2, and CICIDS2017. Since, the focus progressively shifts towards more specific use cases, notably with the generalization of Internet of Things (IoT) devices. These datasets include protocols that are not present in traditional IT-oriented networks, such as MQTT or CoAP. This is the case for Bot-IoT [Kor+19], ToN_IoT [Mou21], and Edge-IIoTset [Fer+22].

2.2.3 Metrics

Most research on ML for intrusion detection relies on the same set of metrics to assess, validate, and compare their solutions [BG16; Cha+19; Far+20]. Most of these metrics are derived from the confusion matrix (see Table 2.2), which is a table that summarizes the performance of a classification model along the different classes. To compute the confusion matrix, the model's predictions ($\hat{y}$) are compared to the true labels (y, the ground truth) of the samples. All the metrics presented in this section are defined for binary classification, but can be extended to multi-class classification [BG16].

(1) *Accuracy* represents the proportion of correctly classified items. It is the ability for the system to correctly distinguish abnormal traffic from legitimate one.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$$

6. Available at: https://www.ntop.org/products/traffic-analysis/nprobe/

		Predicted condition	
	Total population $= P + N$	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True Positive (TP)	False Negative (FN)
	Negative (N)	False Positive (FP)	True Negative (TN)

Table 2.2 – Confusion matrix for binary classification.

(2) *Precision*, or Positive Predictive Value (PPV), is the proportion of correct positive cases among all the cases that have been categorized as positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

(3) *Recall*, or True Positives Rate (TPR) represents the proportion of true positive cases that have been correctly categorized.

$$\text{Recall} = \frac{\text{TP}}{\text{P}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

(4) *Specificity*, or True Negative Rate (TNR), is the proportion of negative cases that has been correctly categorized.

$$\text{Specificity} = \frac{\text{TN}}{\text{N}} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

(5) *Fallout*, or False Positives Rate (FPR), represents the proportion of the positive cases that should have been categorized as negative. A high FPR often requires human intervention after the classification task to filter out the false positive.

$$\text{Fallout} = \frac{\text{FP}}{\text{N}} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

(6) *Missrate*, or False Negative Rate (FNR), relates to the proportion of positive cases that have not been categorized as such. In the context of IDSs, it represents an attack that has been missed by the system. Thus, it is a critical metric for this use case.

$$\text{Missrate} = \frac{\text{FN}}{\text{P}} = \frac{\text{FN}}{\text{TP} + \text{FN}}$$

(7) *F1-Score* is the harmonic mean of precision and recall. It is often used to measure ML algorithm, but is also criticized because of the equal importance it gives to both precision and recall [HC18].

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

(8) *Mathew Correlation Coefficient (MCC)* is an adaptation of the *Phi* (ϕ) coefficient to confusion matrices. While being mathematically identical, the term is often preferred by the ML community. The MCC has significant advantages over the other metrics, as it covers all four categories of the confusion matrix [CJ20]. Thus, a high score can only be obtained with high TP and TN, and low FP and FN.

$$MCC = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TN} + \text{FN}) \cdot P \cdot N}}$$

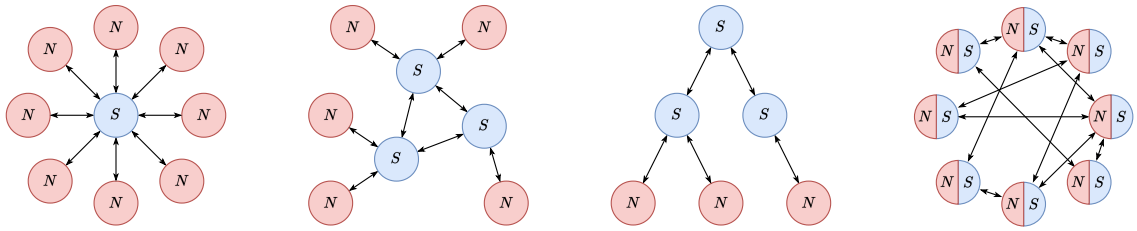
2.3 Collaboration in Intrusion Detection

The topic of collaboration in intrusion detection is rather old, with several surveys and reviews available in the literature [EO11; FS16; LMK22; Men+15; VKF15; ZLK10], and the oldest references dating back to the early 1990s [Sna+92]. In this section, we present the different types of collaboration in intrusion detection, and discuss some of the challenges that they face. A first distinction can be made between their objectives, although they are not mutually exclusive: (i) *share results* to correlate alerts and detect attacks at a global scale, or (ii) *share knowledge* to improve the detection capabilities of local systems.

These objectives also depend on the scale of the collaboration. The first case mostly refers to different probes, or sensors, that monitor the same infrastructure, and share their results to correlate alerts and detect attacks at the infrastructure level. In the second case, which is sometimes referred to as a Collaborative Intrusion Detection Network (CIDN) [LMK22], collaboration usually happens among different organizations or entities that monitor infrastructures. In this thesis, we will focus on the latter, as it is more relevant to the FL context (see Section 2.4).

2.3.1 The different topologies

The aforementioned literature identify two main types of topologies for CIDSs: centralized and decentralized. The definitions of these topologies are not always consistent across the literature, especially between the terms *decentralized*, *hierarchical*, and *distributed*. For instance, Zhou, Leckie, and Karunasekera [ZLK10] consider a decentralized system as a system where each node is autonomous and can make decisions independently, while this



(a) Centralized **(b)** Hybrid **(c)** Hierarchical **(d)** Fully decentralized

Figure 2.4 – Different topologies for collaborative intrusion detection systems. Nodes are in red and marked as N, while servers are in blue and marked as S. Arrows represent connections between entities.

definition matches the description of a distributed architecture in the work of Li, Meng, and Kwok [LMK22].

In this manuscript, we will use the definitions illustrated in Figure 2.4. It distinguishes two types of roles: *nodes* (N) and *servers* (S). A node is a device that captures data, although it can also execute complementary tasks like preprocessing, feature extraction, or traffic analysis. A server is a device that aggregates the data from the nodes and distributes instructions to them, as well as updates for the local detection algorithm. The different topologies are defined as follows, including different levels of decentralization:

Centralized. In a centralized architecture, a single server centralizes knowledge and distributes instructions to the nodes.

Hybrid. In a hybrid architecture, multiple servers coexist. Each server is responsible for a subset of the nodes, and they can share information between them. There are no *central* server per se, but the roles of server and node still exist.

Hierarchical. A hierarchical architecture is a hybrid architecture where the servers are organized in a tree-like structure. Each server is responsible for a subset of the nodes, and they can forward information to their parent. Likewise, parents can distribute instructions and updates to their children so that they are disseminated throughout the hierarchy.

Fully decentralized. In a fully decentralized architecture, each node is autonomous and can make decisions independently. Both roles of nodes and servers coexist in the same entity. There are no servers anymore, and the node share information over a peer-to-peer network.

Figure 2.4 illustrates these topologies. To summarize, the main difference between the different topologies is the level of autonomy of the nodes and the centralization of the knowledge. In the centralized architecture (Figure 2.4a), all nodes are connected to a single server. We can refer to an architecture as decentralized as soon as there are no single server overseeing all the nodes. Figures 2.4b to 2.4d show different examples of a

decentralized architecture. The arrows between the different entities represent information exchange, although the nature of these exchanges can vary depending on the direction. An arrow displayed as $N \rightarrow S$ can represent collected data, generated alerts, or requests for updates. An arrow displayed as $S \rightarrow N$ can represent instructions or updates for the local detection algorithm or database. The term *distributed* refers less to the way the system is organized, and more to how the tasks are executed. In a distributed system, the tasks (*e.g.*, detection, data processing, ML model training) are executed simultaneously on different nodes, which can be organized in any of the aforementioned topologies.

2.3.2 Challenges in Collaborative Intrusion Detection

Collaborative intrusion detection faces challenges, including the Single Point-of-Failure (SPoF) in a centralized architecture. If the analysis is performed remotely, like in a Security Operations Center (SOC) monitoring its consumers' infrastructures, a failure on the central server would hinder detection. Fortunately, in knowledge-sharing scenarios, detection is (at least partially) performed locally, reducing the impact of a centralized failure. Nonetheless, collaboration still relies on the availability of the central server.

Challenge 3. CIDSs typically rely on a central server for coordination and updates, which represent a Single Point-of-Failure (SPoF).

Another challenge in collaborative intrusion detection is the latency induced by propagating information over the network, especially under load. The ENISA (the European Union's agency for cybersecurity) defines the actionability of Threat Intelligence (TI) as the fulfillment of five criteria: relevance, digestibility, accuracy, completeness, and timeliness [ENI14]. It is the supporting architecture that provides the latter. Because low-latency is crucial for actionable alerts locally, centrally analyzing the data increases the time between the event and its detection.

Challenge 4. Centralized detection increases latency, which makes the shared knowledge less actionable.

Further, sharing data can represent a privacy risk for a company, as the data relevant for intrusion detection is likely to contain sensitive information [ZLK10]. Exposed information might reveal relevant insights to a competitor or an attacker.

Challenge 5. CIDSs can expose sensitive information about the internals of a company.

A lot of other factors can impede collaboration. For instance, stakeholders are often reluctant to share their information, fearing confidentiality and privacy issues (see Challenge 5), but most importantly the reputation loss that could result from a breach [PZ19].

Cultural and language barriers can negatively affect the accuracy of the shared information, even though international collaboration is push by regulation, such as the NIS directives in Europe [16; 22]. Finally, the balance between anonymity and trust must be taken into consideration to protect the participants without sacrificing the quality of the information [ML15].

2.3.3 CIDS with Machine Learning

Before the advent of FL, the literature on CIDSs leveraging ML, or more generally data-mining techniques, was scarce [FS16]. Existing solutions were mostly based sharing alerts for correlation or rules for misuse detection, or rely on a central server to perform learning tasks. Nonetheless, a few works leveraged distributed learning techniques that remind of FL, notably with the constraint of not being able to share data between the nodes. For instance, Folino, Pizzuti, and Spezzano [FPS10] proposed a framework allowing distributed IDS nodes to train and exchange classifiers, before aggregating to build ensemble models. However, most of the aforementioned reviews still identify data-mining and ML techniques as a promising direction for CIDSs.

2.4 Fundamentals of Federated Learning

Introduced in 2016 by McMahan *et al.* [McM+17], Federated Learning (FL) changes the usual ML paradigm where distributed data is centrally collected, curated, and processed on a dedicated server. Instead, FL respects the decentralized nature of the data and rather brings model training to the data sources. By alternating between training on local data and aggregating model updates, FL enables the training of a shared model without the need to share the data itself. This approach reveals itself as a promising solution to multiple challenges faced by traditional ML systems. The two main ones are:

1. training models over massively distributed data sources, such as smartphones, wearables, or IoT devices;
2. training models on sensitive data, such as medical records or financial transactions, while preserving privacy and confidentiality.

Although the term *Federated Learning* was introduced by McMahan *et al.* [McM+17] to describe their approach focusing on distributed mobile devices, the literature has significantly broadened the definition of FL to encompass a wide range of privacy-preserving distributed learning techniques. Therefore, we prefer the definition introduced in 2021 by Kairouz *et al.* [Kai+21] and reiterated in Definition 2.1. The following sections introduce the fundamentals of FL, with a focus on the `FedAvg` algorithm, the different types of FL, and the question of data distribution. Finally, we succinctly discuss the threats against

FL, with a focus on data poisoning attacks. Table 2.3 summarizes the notations used in this section, and throughout the manuscript.

Definition 2.1: Federated Learning

Federated Learning is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider. Each clients raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective. – Kairouz *et al.* [Kai+21]

Formally, FL seeks to minimize a objective function $f(\cdot)$ [7], as:

$$\min_{w} f(w), \quad \text{where} \quad f(w) = \sum_{i=1}^{K} \rho_i f_i(w), \tag{2.4}$$

where w is the model parameters, $f_i(w)$ is the local objective function of client i, and ρ_i is the weight of client i.

2.4.1 The `FedAvg` Algorithm

`FedAvg` is the first and most popular algorithm for FL. The algorithm operates in rounds, noted r. At each round r, an orchestrating server S randomly selects $C \cdot K$ clients from a pool of participants P, with K being the total number of participants and C the fraction of clients selected with $0 < C \leq 1$. The server then tasks each selected participant $p_i, i \in [\![1, K]\!]$ to train a model w_i^r. The round ends by the aggregation of the collected models into a new global model $\bar{w}^r$, which is redistributed to the clients as a starting point for the next round $(r + 1)$.

In essence, `FedAvg` is a distributed 2-level SGD algorithm, where C controls the *global* batch size, and then each client p_i uses a *local* batch of size β to compute a local update. To train their model, the participants use a SGD-based optimizer to minimize a objective function $f_i(w)$, which is the local loss function of client p_i (see Definition 2.1). They compute the gradients of the loss function with respect to the model parameters, as:

$$g_i^r = \nabla f_i(w_i^r), \tag{2.5}$$

with is the results of the local optimization process.

7. Note that in the context of intrusion detection using ML, the objective function $f(\cdot)$ is often a loss function $\mathcal{L}(\cdot)$ to minimize, such as mentioned in Section 2.2.1.

Table 2.3 – Summary of Notations.

Notation	Description
K	Number of participants
$P = \{p_i \mid i \in [\![1, K]\!]\}$	Set of all participants
C	Fraction of selected participants
d_k	Local dataset of participant p_k
$D = \bigcup_{i=1}^{K} d_i$	Union of all local datasets
$X_i = \langle x_1, x_2, \ldots, x_m \rangle$	Features of sample i
Y_i	Label enconding for sample i
w_i^r	Local model of the k-ith participant at round r
$W = (w_i^r \mid i \in [\![1, K]\!])$	Local models from participants at round r
$\bar{w}$	Aggregated model at round r
$L(w_i, d_i)$	Loss function for model w_i on d_i
$\mathcal{E}$	Number of local epochs
β	Batch size
η	Learning rate

In their original publication, McMahan *et al.* [McM+17] introduce two algorithms for FL: `FedSGD` and `FedAvg`. `FedSGD` is a straightforward implementation of SGD in a federated setting, where each client computes the gradients after one epoch and sends them to the server. The server then aggregates the gradients and updates the global model. With $D = \bigcup_{i=1}^{K} d_i$ being the union of all local datasets d_i, the server computes the new global model as:

$$w^{r+1} = w^r - \eta \sum_{i=1}^{K} \frac{|d_i|}{|D|} g_i^r, \tag{2.6}$$

where η is the learning rate, and $|d_i|$ and $|D|$ are the sizes of the local dataset and the global dataset, respectively.

Based on the observation that there is no difference between averaging the gradients g_i^r and updating the global model, or updating the model locally and then averaging the results, McMahan *et al.* [McM+17] introduce `FedAvg`. Indeed, the two operations are equivalent, as the following equation shows:

$$w^{r+1} = w^r - \eta \sum_{i=1}^{K} \frac{|d_i|}{|D|} g_i^r = \sum_{i=1}^{K} \frac{|d_i|}{|D|} w_i^r, \tag{2.7}$$

where w_i^r is the model trained by client p_i at round r. This equivalence allows clients to train their models for multiple epochs before sending the results to the server, which reduces the communication overhead. Algorithm 2.1 summarizes the `FedAvg` algorithm.

This core idea, that *averaging locally trained models iteratively converges towards an optimal model trained over distributed data*, is the foundation of FL.

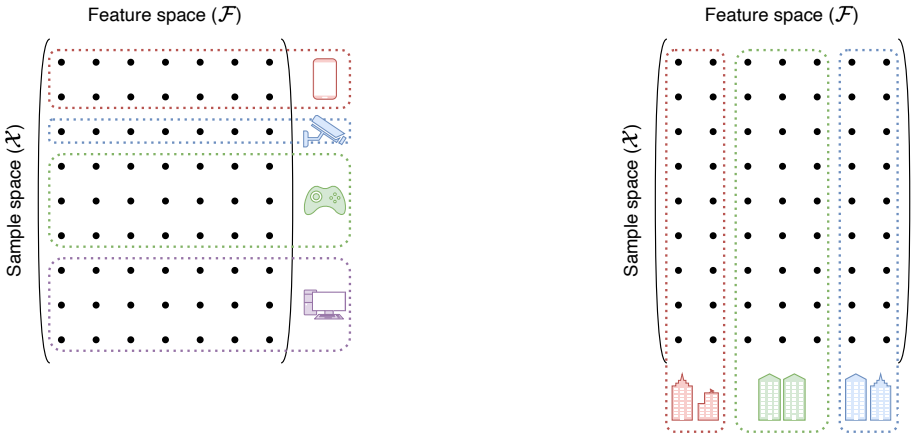
Algorithm 2.1 `FedAvg` [McM+17]. The participants of P are indexed by i, C is the fraction of participants to be selected at each round, β the local batch size, η the learning rate, $\mathcal{E}$ the number of epochs, and $\nabla\mathcal{L}$ the gradients of the loss function $\mathcal{L}$. SPLIT is a function that splits a dataset into batches or size β.

Initialize w_0
for each round $r = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $P^r \leftarrow (\text{SELECTRANDOM}(m, P))$
  for all $p_i \in P^r$ **do**
    $w_i^r \leftarrow \text{CLIENTFIT}(p_i, w^r)$
  $w^{r+1} \leftarrow \sum_{i=1}^{K} \frac{|d_i|}{|D|} w_i^r$
▷ *On client p.* ◁
function CLIENTFIT(p, ω)
  for $i \leftarrow 1, \dots, \mathcal{E}$ **do**
    for all $b \in \text{SPLIT}(d_i, \beta)$ **do**
      $\omega \leftarrow \omega - \eta \nabla \mathcal{L}(\omega, b)$
  return ω

2.4.2 Types of Federated Learning

As stated in the introduction of this section, the literature has broadened the scope of FL, leading to different types of FL depending on the context and the objectives of the federation.

Cross-device *vs.* Cross-silo Federated Learning The first notable distinction is between Cross-Device Federated Learning (CD-FL), the context in which FL was introduced, and Cross-Silo Federated Learning (CS-FL). The *cross-device* settings concerns massively distributed devices which are typically low-power and resource-constrained, such as smartphones, wearables, or IoT devices. Their number can range from thousands to billions, they are often heterogeneous and owned by different users. Consequently, CD-FL often encounters challenges related to limited availability, reliability, and communication overhead, but offers scalability and adaptability. In contrast, in *cross-silo* settings, FL operates within organizational boundaries or distinct data silos, where each silo represents a separate entity or institution. Silos could correspond to different departments within a company, independent organizations, or even geographical regions. CS-FL typically implies organizations with more homogeneous capabilities and more data to train on. Parties in cross-silo FL are more likely to be reliable and consistently available for participation, as they are usually institutional entities with dedicated infrastructure and resources. Yet, entities involved in CS-FL also tend to have considerably greater discrepancies in terms



(a) Horizontal Federated Learning (HFL) **(b)** Vertical Federated Learning (VFL)

Figure 2.5 – Horizontal *vs.* Vertical Federated Learning. In horizontal FL, clients share the same features but not the same samples. In vertical FL, clients share the same samples but not the same features.

of objectives and data-distributions, and sometimes even model architectures.

Horizontal *vs.* Vertical FL Another major distinction in FL is the axis along which the data is distributed. In most application (notably in CD-FL), participants share the same features, but possess different samples. This is referred to as Horizontal Federated Learning (HFL) by Yang, Liu, *et al.* [Yan+19], and is illustrated in Figure 2.5. HFL particularly copes with *ground-truth* issues (Challenge 2) by providing more data for the global model to be trained on. Conversely, in Vertical Federated Learning (VFL), participants might have different views over the same data, *i.e.* they share the same samples but not the same features. This is particularly relevant in cross-silo applications, where different organizations might have access to different data sources, but observe the same events. Finally, Yang, Liu, *et al.* [Yan+19] also consider Federated Transfer Learning (FTL), where participants share only a subset of both, features and samples.

Architecture Discrepancies In light of the architectures presented in Figure 2.4, the initial FL proposal [McM+17] would be considered as a distributed task (*i.e.*, model training) that is *centrally* orchestrated (see Figure 2.4a). The central server plays a pivotal role in distributing model parameters, orchestrating training rounds, and aggregating updates from individual devices. This approach requires global coordination and synchronization, as all communication and aggregation activities are orchestrated by the server. While server-orchestrated FL offers centralized control and streamlined management, it also introduces potential single points of failure and scalability limitations due to the server's

central role (see Challenge 3). Although FL's original definition implies a client-server architecture, the literature has also explored other settings. Researchers have explored multiple alternatives to the server-orchestrated setting, such as hierarchical FL [Liu+20] or fully decentralized FL using gossip algorithms [Tan+23]. This decentralized approach eliminates single points of failure and allows for greater scalability, as communication and computation can be distributed across numerous devices. However, fully decentralized FL may face challenges related to coordination, consistency, and synchronization, especially in scenarios with a vast number of participating devices.

2.4.3 The Question of Data Distribution

The performance of FL algorithms is highly dependent on the distribution of the data across the participants. Almost by definition, data in FL settings is Non Independent and Identically Distributed (NIID), as it is distributed across different devices or organizations, with no guarantee of homogeneity. However, the performance of FL algorithms of the literature is often evaluated under the assumption that the data is Independent and Identically Distributed (IID), which is rarely the case in practice. This discrepancy between the theoretical assumptions and the practical reality poses a significant challenge for the FL community.

In the FL foundation paper [McM+17], the authors emphasize on NIID data being one of the key attributes of FL, alongside the unbalanced overall distribution. They notably present a *pathological*-NIID situation using MNIST [Lec+98], a digit recognition dataset, where each client is given only two digits, *e.g.* 3 and 7. More recent papers consider alternative NIID use cases, deemed more realistic. For instance, Huang *et al.* [Hua+21] present a *practical*-NIID use case, where participants can share similarities. This is particularly suited for cross-silo use cases, such as CIDSs. Indeed, we can easily expect different organizations to own different architectures, and yet observe similar traffic patterns in their networks.

The literature has addressed the issue of NIID data in FL from multiple angles. First, some algorithms have been specifically designed to handle NIID data, such as `FedProx` [Li+20c], `Fed+` [Kun+22], or `SCAFFOLD` [Kar+20], although the former also covers the topic of heterogeneous capabilities. Techniques such as client-side sampling, in which clients sample their data to match the global distribution, have also been proposed [Han+24]. Finally, the literature has also explored clustering approaches to group client with model updates in communities, assuming that similar updates come from clients with similar data distributions [Ye+23].

2.4.4 Threats against Federated Learning

The distributed nature of FL opens the way to various attack vectors, which can be classified into two main categories: attacks against the federated model and attacks targeting the participants' privacy. In the former, adversaries aim to alter the behavior of the global model, either to degrade its performance or to manipulate specific predictions. In the latter, adversaries seek to infer sensitive information about the participants' data, such as inferring the presence of a specific sample in a participant's dataset.

The two categories obviously have different objectives, and consequently different threat models. We focus on the former in this thesis. Authors often refer to poisoning attacks in FL as *Byzantine* attacks, as they are analogous to the concept of Byzantine faults in distributed systems. Likewise, the term *Sybil attacks* [Dou02] is frequently used to refer to the problem of *colluding attackers* [FYB20].

Attack vectors Poisoning attacks can be categorized into two main categories depending on the phase in which they are perpetrated: model-poisoning [Bha+19] or data-poisoning [Tol+20]. Model-poisoning attacks aim at manipulating the model's parameters, usually during or after training, to deviate the aggregated model from the global optimum [Fan+20b]. Data-poisoning attacks, on the other hand, happen before the training phase, and manipulate data samples to degrade performance, cause misclassification, or introduce backdoors [Rod+23].

Data poisoning attacks can be categorized into clean-label and label-flipping attacks. Clean-label attacks manipulate the samples to be misclassified, either by adding new samples [Zha+22a] or by modifying existing ones [Mer+23]. Label-flipping attacks, on the other hand, change the labels of the samples by flipping them to a different class [Tol+20]: *i.e.*, $y_{\text{source}} \rightarrow y_{\text{target}}$.

Attack target Additionally, most poisoning attacks can be further separated into *untargeted* and *targeted* attacks. Untargeted attacks randomly select samples to be manipulated, and are usually easier to detect as they have a higher impact on the model's performance. Targeted attacks, on the other hand, select samples based on a specific criterion, such as the class to be targeted. In a CIDS context, targeted attacks can be used to introduce backdoors—*i.e.*, making a specific attack class be misclassified as benign—or cause targeted misclassification.

Mitigation strategies Algorithmic solutions to mitigate these attacks exist in distributed learning, such as Krum [Bla+17] or Trimmed Mean [Yin+18], and are often used as comparison for works in Byzantine-robust FL. In addition to the algorithmic countermeasures, various strategies have been proposed to detect and mitigate poisoning attacks in FL specifically, ranging from clustering [Ngu+22; STS16] and similarity-

analysis [ALL21; FYB20] to client-side evaluation [Zha+20c]. Chapter 6 will provide a more in-depth overview of the state-of-the-art in Byzantine-robust FL.

2.5 Conclusion and takeaways

In this chapter, we have introduced the basics of ML for intrusion detection and the challenges of scaling up to CIDSs. We have also introduced the fundamentals of FL its implications. With the challenges listed along the way, the relationship between FL and CIDS becomes straightforward: FL is a natural fit for CIDSs as it allows to leverage the benefits of distributed learning while preserving the locality of the data.

In the next chapter, we will review the state of the art in FL in the context of CIDS, focusing on the different approaches to the problem and the challenges they face. Notably, we will discuss **??** RQ1: *What makes applying FL to IDSs specific?*.

Chapter 3

Systematic Literature Review and Taxonomy

Contents

3.1 Introduction and Motivation

In the previous chapter, we introduced the concepts of Intrusion Detection System (IDS) and Machine Learning (ML), the challenges of deploying Collaborative Intrusion Detection Systems (CIDSs), and why Federated Learning (FL) is a promising solution to these challenges. This chapter's prime objective is to provide a comprehensive review of how Federated Learning (FL) can be leveraged for intrusion detection purposes, and shed light on the gaps in the literature that are discussed in this thesis.

A recent topic without identity Because of the novelty of FL in the field of Intrusion Detection System (IDS), the literature on the topic is still scarce. Only a handful of reviews [Agr+22; Ala+21; Cam+22] existed on the topic when we stopped our data collection for this study in late 2021, most of which only as preprint. While these papers provide a good overview of the existing works, they fail to provide synthesis and extract the core characteristics of the field. Notably, *what makes FL for IDS different from FL for other applications, and what challenges are specific to the field of intrusion detection?*

A systematic approach We aim to address this gap as thoroughly and transparently as possible, and leverage the Systematic Literature Review (SLR) methodology to that end. This method relies on a structured process to identify, select, and analyze the relevant

literature on a given topic. With explicitly defined research questions and inclusion/exclusion criteria, the Systematic Literature Review (SLR) methodology ensures that the review is reproducible and unbiased. Therefore, we intend to provide a comprehensive overview of the existing literature, and reproducible, evidence-based conclusions on the specificities of FL for IDS.

Content The content of this chapter is based on our survey published in IEEE Transactions on Network and Service Management (TNSM) in May 2022 [Lav+22b] and its accompanying extension at the C&ESAR conference in November 2022 [Lav+22a]. Because the initial paper was submitted in November 2021, the quantitative analysis has been updated during the writing of this manuscript to include the latest publications on the topic. The qualitative analysis has also completed to a lesser extent.

Contributions of this chapter

- The first SLR on the use of FL for IDS, including qualitative and quantitative analyses of the literature.
- A generalization of the selected works as a reference architecture for Federated Intrusion Detection Systems (FIDSs), providing a starting point for future works on the topic.
- A taxonomy synthesizing the state of the art of FIDS, providing a framework to analyze and compare existing and upcoming literature.
- The identification of the main challenges and opportunities in the field, and a set of research directions to address them.

3.2 Methodology

This section details the methodology applied to review the state of the art of FIDSs. The SLR methodology was originally introduced to the field of engineering by Kitchenham and Charters [KC07]. SLR uses analytical methods to answer research questions about the literature on a specific topic. The update to the original article is less structured and more focused on the evolution of the field, so the methodology is adapted accordingly.

3.2.1 Research Questions

The SLR methodology recommends defining explicit research questions to structure the review and the selection of papers. This survey aims at evaluating FIDS and their maturity, as well as their core components, and relevant variations. Therefore, using re-

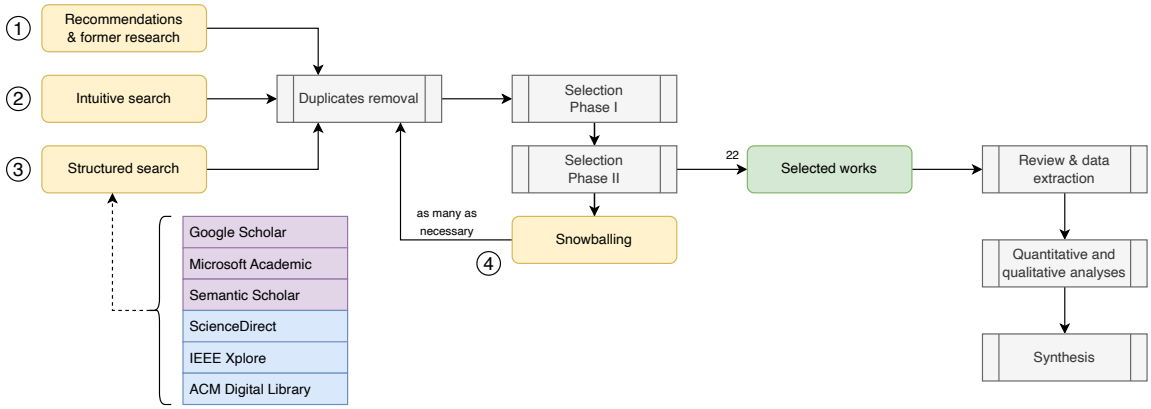


Figure 3.1 – *Search and selection processes.* Sources of papers appear in yellow, the final selection in green, and the processing steps in gray. The tools used in the *Structured search* are presented with search engines in purple, and online databases in blue. Figure from Lavaur, Pahl, *et al.* [Lav+22b] © IEEE 2022.

lated and selected works, we identify the following Research Questions (RQs) that cover the topic of FIDSs. The questions complete and extend **??** RQ1 which was introduced in Chapter 1.

RQ1-1. *What are FIDS?*

- **1.a.** What challenges do FIDS help to cope with?
- **1.b.** Which techniques exist to federate Machine Learning (ML)–based detection and mitigation mechanisms?

RQ1-2. *What are the differences between FIDS?*

- **2.a.** What are the key components of FIDS? How do they influence the system's performance?
- **2.b.** Which metrics are used to measure and compare FIDS?

RQ1-3. *What is the state of the art of FIDS?*

- **3.a.** What are the subtopics covered by the academic literature since 2016?
- **3.b.** Where was the literature published? Which research groups and communities are active in this area?
- **3.c.** What are open questions according to existing works?

3.2.2 Search and Selection Process

Figure 3.1 presents the methodology and its search, selection, and synthesis processes. The searching of relevant literature involves four sources: recommendations, intuitive search, structured search, and snowballing.

(1) *Recommendations* were given by supervisors and coworkers throughout the realization of this work. This initial set of relevant papers is also used as a source of snowballing for further searching.

(2) *Intuitive search* has been performed at the beginning of the survey to get a first grasp on the topic, and to learn about the functioning of FIDSs. At first, mostly Google Scholar has been used.

(3) *Structured search* has been adopted afterward, following the principles of SLR [KC07]. Different search engines and online databases are used for the sake of completeness, as illustrated in Figure 3.1. Both can provide different results, depedning on their ownership and scope, as well as the way papers are indexed. Thus, multiple sources provide more exhaustive results. The following queries have been used to search for relevant literature: (a) application of FL to IDSs, and (b) literature addressing the topic of FIDS with unusual keywords.

(a) `("federated learning" OR "fl" OR "federated")`
`AND ("intrusion detection systems" OR "ids")`

(b) `("federated" OR "collaborative")`
`AND ("detection" OR "defense" OR "mitigation")`

(4) *Snowballing* identifies relevant works that would have been missed otherwise, such as publications cited by articles of our selected corpus, or papers that refer to them. The related surveys identified in this work (Section 3.3) contain a lot of references to technical articles, making them relevant for snowballing. Furthermore, as this survey proceeds with quantitative analysis of the venues and groups (Section 3.4), it provides extended snowballing opportunities by looking at other publications in the most represented venues or research groups in the selected corpus.

Approximately two hundred papers have been identified. Duplicate removal is performed with Zotero which allows identifying and merging redundant items. The selection then happens in two phases. Firstly, the title and abstract are used to discriminate *out-of-scope* papers in Phase I, along with their number of citations given the search engines, and age. However, a paper with few citations, but interesting abstract, probably only lacks visibility. Thus, it is moved to Phase II, which consists of a more thorough analysis of the selected works, using the *three-pass* approach defined by Keshav [Kes07].

After the two selection phases, 22 papers were selected, excluding the 18 initial surveys seen in Section 3.3. All present technical solution for FIDS. The challenges identified in Chapter 2 were also used to either search or select papers, mostly through the *intuitive search* part.

3.2.3 Data Extraction and Analysis

The quantitative section of the original paper was solely based on the 22 selected papers. However, a significant amount of literature has been published since the initial survey. Therefore, we updated the quantitative analysis to include the latest publications on the topic. The qualitative analysis has also been completed to a lesser extent, just

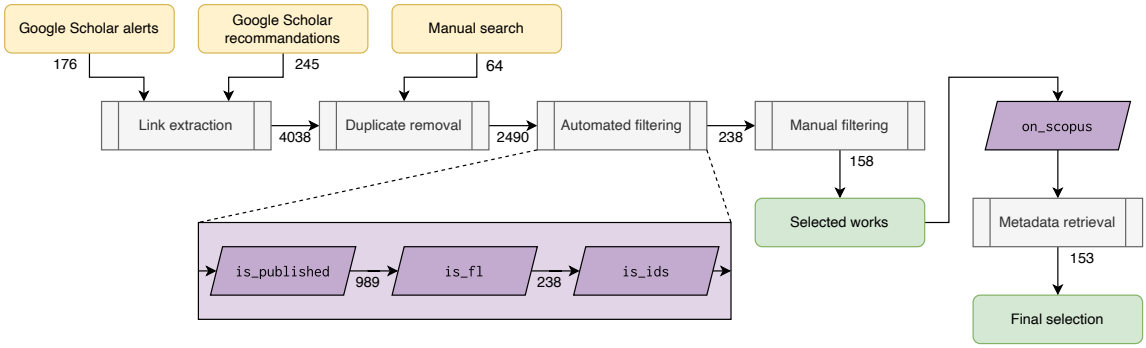


Figure 3.2 – *Updated selection process.* The sources of papers are in yellow, the selected papers in green, and the processing steps in gray. Purple parallelograms the automated filters used to select papers.

enough to provide a general overview of the field's evolution. Figure 3.2 presents the methodology applied to update the presented results.

We set up an automated collection system on Google Scholar, composed of an alert based on the search queries defined in Section 3.2.2 and automated recommendations. The system was set up in 2021 and ran until the beginning of the writing of this manuscript in April 2024. It brought 423 emails containing 2490 links after duplicate removal. A first selection was performed on the title and abstract, yielding 238 papers. After manual filtering, we select 158 relevant papers, which amount to 136 new publications since the original survey.

Literature processing To process this new corpus, we use Litstudy [Hel+22], a Python library providing tools to extract and analyze bibliographic data. On the 158 selected papers, 153 only were available on Scopus, the database available in Litstudy that provides the most complete data. The list of papers is available as appendix of this manuscript. The data extracted from the papers includes the title, abstract, authors, publication venue, publication date, and keywords, among others. The extracted data is then used to perform a quantitative analysis of the literature presented in Section 3.4.

Topic modeling Litstudy also provides tools to perform topic modeling on the text data of the papers, mainly title and abstract. We first preprocess the text data by removing stop words, punctuation, and numbers, and then associate each word with its frequency in the corpus. Then, we test the two main approaches of available in the literature, namely Latent Dirichlet Allocation (LDA) and Non-negative Matrix Factorization (NMF), to identify the main topics in the corpus. The presented results have been obtained using the NMF algorithm on 20 topics and after 2000 iterations, as it provided the most interpretable results.

Table 3.1 – *Related literature reviews, their topics, contributions, and number of citations according to Google Scholar (Apr. 2024).* Works marked ∗ were originally available as preprints, and were only published afterward. Works marked ‡ are added for the sake of completeness, but were not included in the initial selection.

Domain	Year	Authors	Contributions							Cited	Ref.
			Qualitative analysis	Quantitative analysis	Taxonomy	Reference architecture	Performance evaluation	Research directions	Systematic Literature Review		
Security information sharing	2016	Skopik *et al.*	●	○	○	○	○	●	○	291	[SSF16]
	2018	Tounsi *et al.*	●	●	○	○	○	●	○	448	[TR18]
	2019	Wagner *et al.*	●	●	○	○	○	●	○	240	[Wag+19]
	2019	Pala *et al.*	●	●	○	●	○	●	○	63	[PZ19]
ML for intrusion detection	2016	Buczak *et al.*	●	○	○	○	○	◐	○	3105	[BG16]
	2018	Meng *et al.*	●	○	○	○	○	●	○	562	[Men+18]
	2019	Chaabouni *et al.*	●	○	◐	○	○	●	○	790	[Cha+19]
	2019	da Costa *et al.*	●	○	○	○	○	●	○	492	[dCos+19]
Collaborative detection	2010	Zhou *et al.*	●	○	○	○	○	●	○	517	[ZLK10]
	2015	Vasilomanolakis *et al.*	●	○	●	○	○	●	○	379	[VKF15]
Federated learning	2020	Aledhari *et al.*	●	○	○	○	○	○	○	517	[Ale+20]
	2020	Lyu *et al.* ∗	●	○	○	○	○	●	○	436	[LYY20]
	2020	Shen *et al.*	●	○	○	○	○	●	○	69	[She+20]
	2021	Mothukuri *et al.*	●	○	●	○	○	●	○	376	[Mot+21a]
	2021	Lo *et al.*	●	●	○	○	○	●	●	158	[Lo+21]
FL for intrusion detection	2021	Agrawal *et al.* ∗	●	○	○	○	○	●	○	142	[Agr+22]
	2021	Alazab *et al.*	●	○	○	○	○	●	○	158	[Ala+21]
	2021	Campos *et al.* ∗	●	○	○	○	●	●	○	123	[Cam+22]
	2022	**Lavaur *et al.***	●	●	●	●	○	●	●	22	[Lav+22b]
	2022	Fedorchenko *et al.* ‡	◐	○	○	○	○	○	○	22	[FNS22]
	2022	Ghimire *et al.* ‡	●	○	○	○	○	●	○	208	[GR22]
	2024	Isma'ila *et al.* ‡	●	●	○	○	○	●	●	0	[Ism+24]

● covers topic; ◐ partly addresses topic; ○ does not cover topic.

3.3 Related Work

At the time of writing this literature review, the literature on FL for IDS was still scarce. Only a handful of reviews had been published on the topic [Agr+22; Ala+21; Cam+22]. Therefore, we extended our search of related works to related topics that were susceptible to share similar challenges or conclusions. This extended selection can be divided into three main categories: (a) security information sharing, (b) intrusion detection, and (c) collaborative ML. Table 3.1 provides a summary of this selection, grouped by topic and sorted by publication date. In addition to the initial selection, we also included more recent surveys on the topic [FNS22; GR22; Ism+24], whose number highlights the massive interest in the community.

Common issues of collaborative systems, such as the need for trust, privacy, and security, can also apply to FL-based collaboration systems. Therefore, we include four surveys [PZ19; SSF16; TR18; Wag+19] where the authors discuss the challenges and opportunities of sharing security-related information. They highlight the need for standardization, automation, and incentives, to achieve efficient and effective collaboration. The topic of trust is a clearly identified challenge in these works [TR18; Wag19]. The present study rather focuses on FL as a technical mean for collaboration, but such as trust or incentives are also relevant in this context.

Because ML-based IDS can be considered as a key component of FIDS, we review existing surveys on the topic [BG16; Cha+19; dCos+19; Men+18]. These work cover a wide range of solutions, from traditional ML (Support Vector Machine (SVM), Decision Tree (DT) and Random Forest (RF), among others) to more recent approaches, such as deep learning, the latter being overrepresented in the literature of FIDSs. They also provide a good overview of the existing datasets and evaluation metrics, which can be useful for the evaluation of FL-based IDS. However, as noted in Section 3.6.2, typical IDS datasets present limitations that can hinder the evaluation of FL-based IDS.

FL's performance is obviously another critical aspect of FIDSs. Consequently, related works include surveys on the collaborative aspects of ML and FL [Ale+20; Lo+21]. They discuss FL approaches to work with distributed architectures. The security of FL is also heavily reviewed by [LYY20; Mot+21b; She+20]. They identify security threats like communication bottleneck, poisoning, and Distributed Denial of Service (DDoS) attacks, that could endanger FL-based systems. While the IDS use case can be seen as an application of FL, we argue that it raises specific concerns in terms of privacy, latency, and adaptability.

Vasilomanolakis, Karuppayah, and Fischer [VKF15] and Zhou, Leckie, and Karunasekera [ZLK10] survey the evolution of Collaborative Intrusion Detection System (CIDS)—at the merge of intrusion detection and collaborative ML, or Topics (b) and (c) as presented above. Their works are however older and thus, cannot offer a comprehensive view of CIDS, as FL-based approaches did not exist at the time of their writing. Hence, the authors focus on collaboration in the sense of *detection+correlation*, whereas the analysis presented in this chapter (Section 3.5) surveys the use of FL in IDSs.

In addition to the above, recent work (*i.e.*, contemporary to the writing of the initial study) have reviewed the use of FL for intrusion detection [Agr+22; Ala+21; Cam+22]. Alazab *et al.* [Ala+21] address the wider topic of FL for cybersecurity, which only includes intrusion detection as an application. Their paper is explanatory and provides an overview of FL applications in information security. Like this work, Agrawal *et al.* [Agr+22] focus on FIDSs, but have different methodology. The authors list existing FIDSs and detail their approaches, and identify open issues. On the other hand, Campos *et al.* [Cam+22] review a subset of FIDSs by focusing on Internet of Things (IoT) use case, and the impact of non-IID (Independent and Identically Distributed) data on performance. While all identify

challenges and research directions, this work also performs quantitative (Section 3.4) and qualitative (Section 3.5) analyses of existing FIDSs, and extracts reference architecture and taxonomy. The existence of these papers emphasizes the importance and relevance of FIDSs for the research community.

The more recent works on the topic [FNS22; GR22; Ism+24] confirm these observations. The work of Fedorchenko, Novikova, and Shulepov [FNS22] is of little interest, as it only lists and details existing works with close to no added value. Ghimire and Rawat [GR22] provide a more convincing study, closer to the method applied by Alazab *et al.* [Ala+21], but with a focus on the IoT. Finally, Isma'ila *et al.* [Ism+24] provide a comprehensive review, with up-to-date literature leveraging the SLR methodology, but still focuses on the IoT.

3.4 Quantitative Analysis

This section provides indicators of the representation of FIDSs in the scientific literature: the evolution of the publications, the relevant venues, the active groups, and the topics of interest. Notably, the identification of the most active groups and most relevant venues provides insights on how to keep track of the most recent advances in the field.

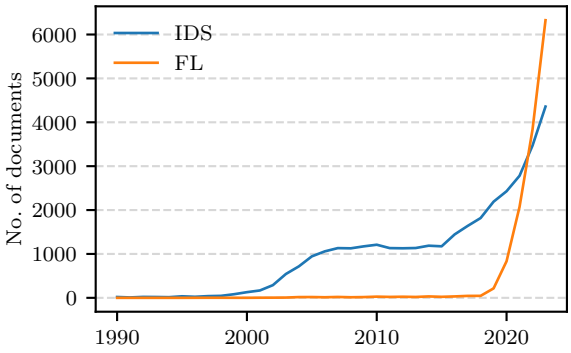
3.4.1 Evolution of the Topic

The topic of IDS started to gain traction in the late 1990', as depicted in Figure 3.3a. After a stagnation period, the topic regained interest around 2015, with an increase of the research on IoT and Industrial Internet of Things (IIoT) [Cha+19; DAF18], alongside other specific use cases. With the introduction of FL by McMahan *et al.* [McM+17], the community started to explore the application of FL to IDS around the years 2018–2019. Figure 3.3a has been generated using the analytics offered by Scopus and the following queries:

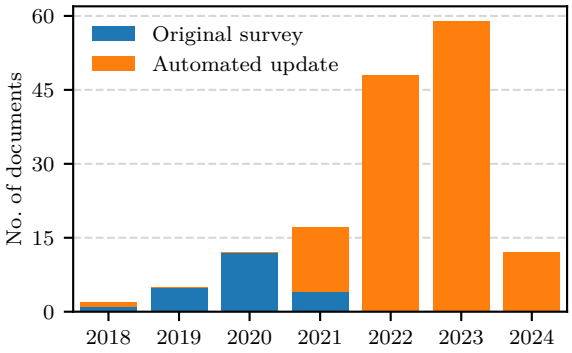
(a) `intrusion AND detection AND system`;

(b) `federated AND learning`.

Recent works on FL focus on its security and privacy-preserving aspects [LYY20; Mot+21b; Ngu+20]. Techniques like homomorphic encryption were introduced as early as 2017 [Har+17], and have been extensively reviewed since. More recently, other privacy-preserving techniques have been applied to FL, such as Multi-Party Computation (MPC) in FLGUARD [Ngu+21] or differential privacy in [KGS21]. FIDSs present a similar tendency with more research towards algorithm security and privacy-preserving techniques. For instance, Li, Wu, *et al.* [Li+20a] use homomorphic encryption to provide a secure and privacy-preserving aggregation of models. Aside from security, variations of Horizontal



(a) Evolution of the topics using Queries (a) and (b) according to Scopus, up to 2023.



(b) Evolution of the number of publications on FIDSs (see Section 3.2.3).

Figure 3.3 – Evolution of the topics and number of publications.

Federated Learning (HFL) started to appear in 2021, such as segmented FL in [Sun+20], as standard HFL has significantly been studied for FIDS.

Finally, the numerous literature reviews published since 2021 [Agr+22; Ala+21; Cam+22; FNS22; GR22; Ism+24; Lav+22b] show the continuous interest of the community for the study of FIDSs. These also show the need for synthesis and structuring of research in this area.

3.4.2 Relevant Venues

The initial study published in 2022 [Lav+22b] observed very few recurring venues for the publication of FIDS research. Indeed, only three venues had more than one publication on the topic: the *IEEE Internet of Things Journal* [Pop+21b; Zha+20a], *IEEE Access* [Che+20; Li+20b], and the *IEEE BigData* conference [Cet+19; Fan+20a]. The original distribution in terms of venue type (11 conferences, 10 journals and 1 book chapter) has significantly changed, since journals represent two thirds of the publications. It is worth noting that the number of publications in conferences is twice inferior to the number of publications in journals, as depicted in Figure 3.4. Multiple reasons can explain this shift, such as a gain in maturity of the field. However, it is likely that the COVID-19 pandemic partially influenced this trend, as conferences were more impacted by the restrictions than journals.

Another observation of the initial study was the diversity of the venues, spanning a wide range of topics, from IoT to Industrial Control System (ICS), including transportation systems and extra-terrestrial networks. This diversity is still present in the most recent publications, although a few generic venues now host significantly more publications: the *IEEE Internet of Things Journal*, *IEEE Access* and *Computers & Security*. The latter is the first security-specific venue to appear in the list. Its place in the top venues is a sign of the increasing interest of the security community for FL and FIDS,

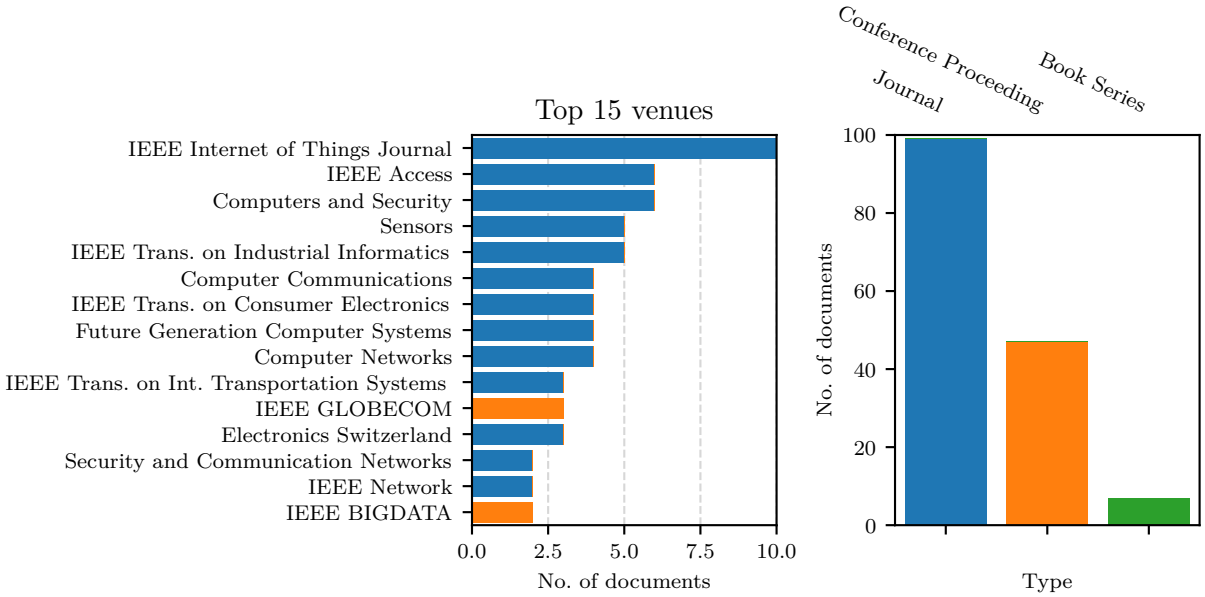


Figure 3.4 – Distribution of the publications in the most recurring venues.

as most contributions were previously published in more use-case specific venues. Lastly, while relevant venues have been accepting FL literature since its introduction, they start to host specific tracks or special issues, such as ICDCS's track on "Federated Learning, Analytics, and Deployment", or IEEE BigData's "Special Session on Federated Learning on Big Data".

3.4.3 Active Groups

Since they introduced the topic of FL in 2016, the team at Google Research has been a big influence for the research community [Bon+17; Bon+19; Kon+16a; Kon+16b; McM+17]. They mostly work on the primitives behind FL, such as model aggregation with the `FedAvg` algorithm [McM+17]. The team of TU Darmstadt (Germany) has also been very active in the field, with a focus on IDS with `DÏoT` [Mar+19; Ngu+19] and FL security [Ngu+20]. The two collaborated, bringing `FLGUARD` [Ngu+21] and `FLAME` [Ngu+22], two algorithms focusing on limiting the impact of poisoning attacks in FL. These series of works makes them one of the most impactful groups in the field.

Other noteworthy groups include the Aalto University (Finland) [Ngu+21] and the University of Tokyo (Japan) [QK21; SEO21; Sun+20]. The most active country remains China, with a dozen institutions now amounting to a third of the publications in the field, as illustrated by Figures 3.5 and 3.6b.

Investigating the major authors tells another story, as the most active authors are not necessarily affiliated with the groups mentioned above. In particular, Popoola, Gui, *et al.* co-authored several publications on FIDS [Pop+21a; Pop+21b; Pop+22; Pop+23] as a collaboration between the Nanjing University of Posts and Telecommunications and

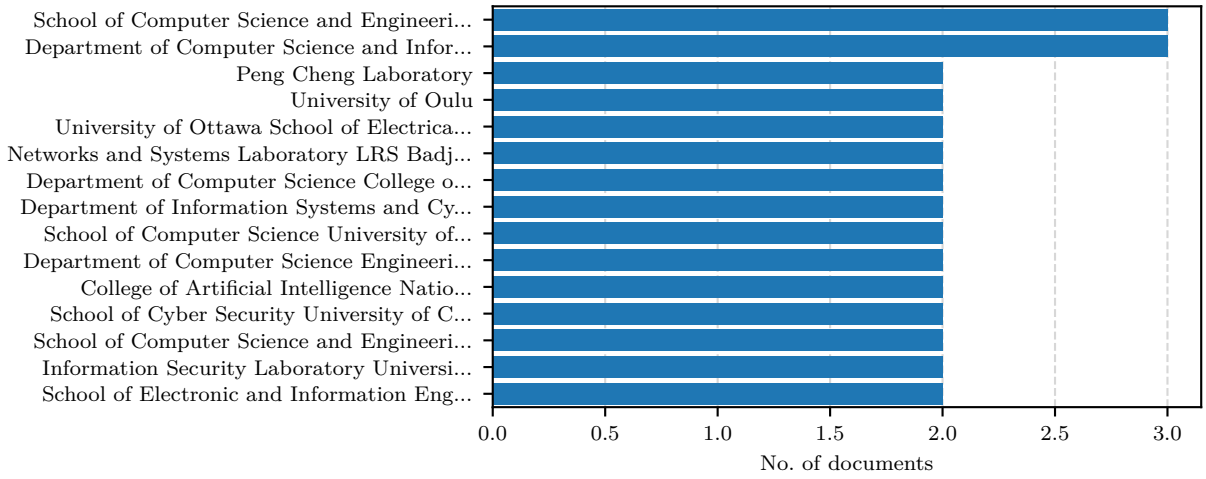
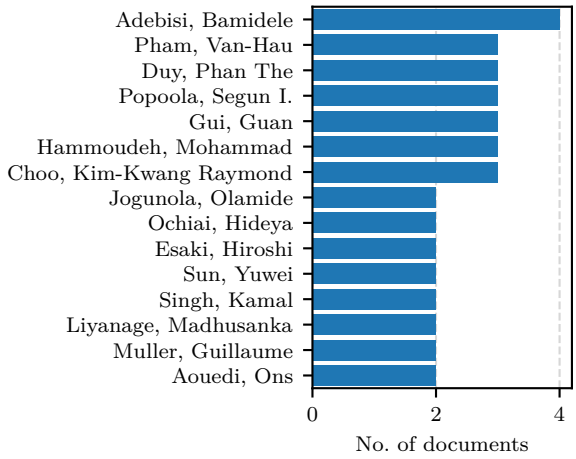
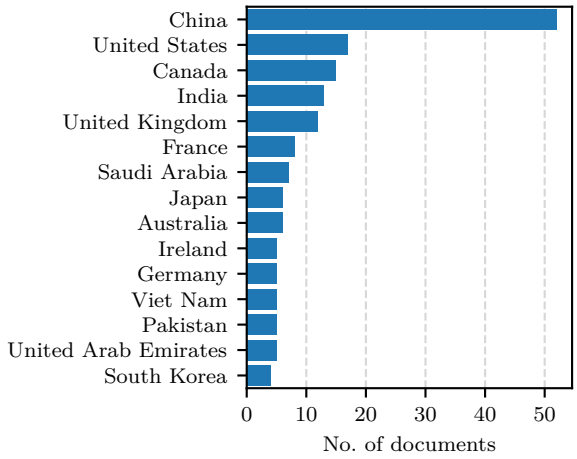


Figure 3.5 – Distribution of the publications by affiliation.



(a) Publications by author.



(b) Publications by country.

Figure 3.6 – Distribution of the publications by author and country.

multiple British universities. Likewise, Duy *et al.*, from the University of Information Technology (VNU, Vietnam), are also quite represented in terms of publications [Duy+21; Quy+22; Thi+22; Vy+21].

3.4.4 Topics of Interest

Using topic modeling, we extract the main topics of interest from the 153 publications on FIDSs identified in the updated selection. By construction, the model is unable to differentiate between application domains (such as IoT or ICS) the techniques used (*e.g.*, blockchains) or the addressed challenges in a paper. However, it provides a good overview of the main topics of interest in the field, especially for the consequent amount of literature published since the initial study. Figure 3.7 present the topics identified by the model, with the most recurring keywords for each topic.

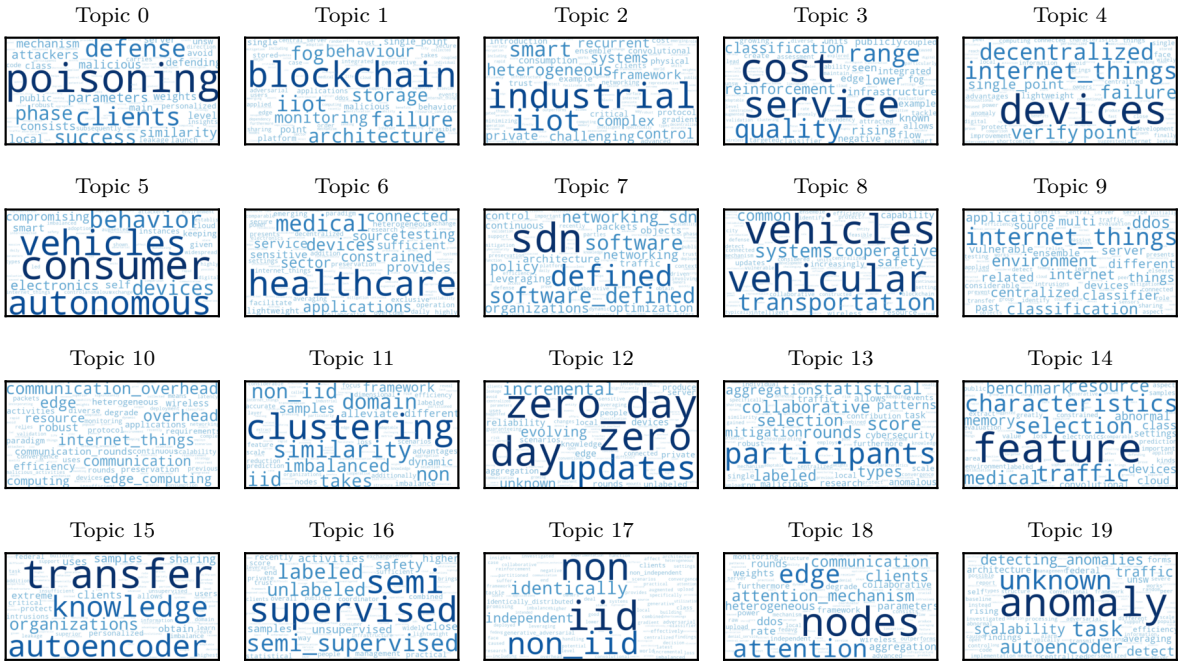
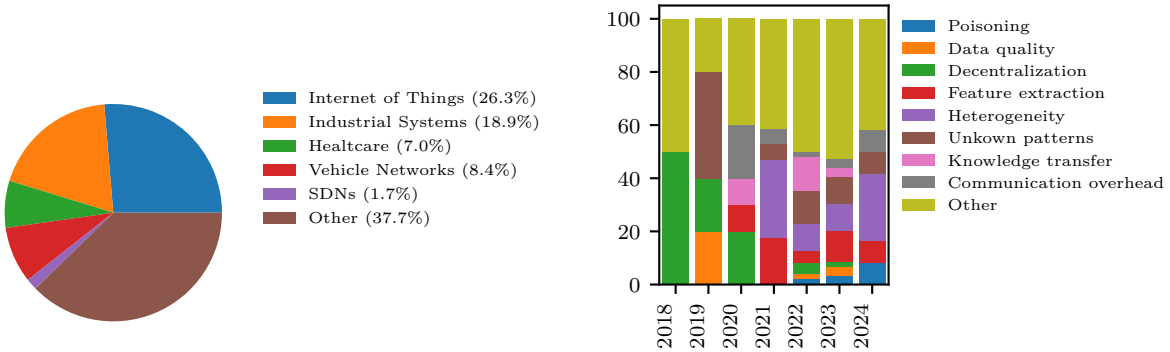


Figure 3.7 – Topics of interest in the field of FIDSs.

First, this analysis highlights the application domains of FIDSs, where the topic of IoT (*i.e.*, `internet_things`, `edge`, `things`) is one of the most recurring (Topics 4, 9, and 10). Other applications stand out, such as ICS (`industrial`, `iiot`), Internet of Medical Things (IoMT) (`medical`, `healthcare`), Vehicle-to-Everything (V2X) (`vehicle`, `vehicular`, `transportation`), and Software-Defined Networking (SDN) (`software`, `defined`, `sdn`). These applications also correlate with the venues identified in Section 3.4.2, as the *IEEE Internet of Things Journal* or the *IEEE Trans. on Industrial Informatics* do focus on IoT and ICS, respectively. Figure 3.8a depicts the distribution of the publications by domain overall.



(a) Distribution of the publications by domain.

(b) Distribution of the addressed challenges over time.

Figure 3.8 – Exploiting the topics of interest.

Likewise, some topics are directly associated with the challenges identified in Section 3.6.2. For instance, Topic 0 (`poisoning`, `defense`, `malicious`) represents works focusing on adversarial attacks against FIDSs and their mitigation. Some techniques can also be extracted from these results. For instance, Topic 0 also contains `similarity` as a keyword, which is likely to refer to the use of similarity metrics to detect poisoning attacks. This is indeed one of the most represented mitigation techniques in the literature on FIDS [Yan+23] or FL alike [FYB20; Ngu+22]. Figure 3.8b depicts the distribution of the addressed challenges over time. Unlike the distribution of the publications by domain, some challenges are addressed in the literature much later, such as handling the heterogeneity of the data (**??**) or resisting to adversarial attacks (**??**). Both are challenges that have been identified in the initial study as open issues in the field [Lav+22a; Lav+22b].

3.5 Qualitative Analysis

This section contains the results of our literature review. First, it synthesizes the analyses into a reference architecture and a taxonomy for FIDSs, which help structure the field. Then, it goes over a comparison of the selected works to answer Questions RQ1-2.a to RQ1-1.b on the components of FIDSs and their impact on performance.

3.5.1 Structuring the Literature

The quantitative (Section 3.4) and qualitative (Section 3.5) analyses provide results that we synthesize hereafter in a reference architecture and a taxonomy. The reference architecture presents the components of FIDSs and their interactions, while the taxonomy provides comparison criteria for the selected works.

We build the taxonomy upon different existing ones related to CIDS [VKF15; ZLK10], ML–based intrusion detection [dCos+19], and FL [Ale+20; LYY20; Mot+21b]. First, we extract classes relevant to the domain of FIDS, before filtering out irrelevant ones by validating the taxonomy against the reference architecture (Figure 3.9). The latter displays both the operation and the design of the system. By confronting the taxonomy and the architecture, we ensure that each item of the taxonomy is related to a component of the architecture, and *vice versa*. Then, we add any commonalities between the selected works that are not already represented in the previous taxonomies. This identifies new criteria on which to compare the selected works.

Reference Architecture

This section presents the reference architecture synthesized from the selected works, as depicted in Figure 3.9. The architecture provides a summary of the components of FIDSs and their interactions, answering Question RQ1-2.a. It can be divided in three parts:

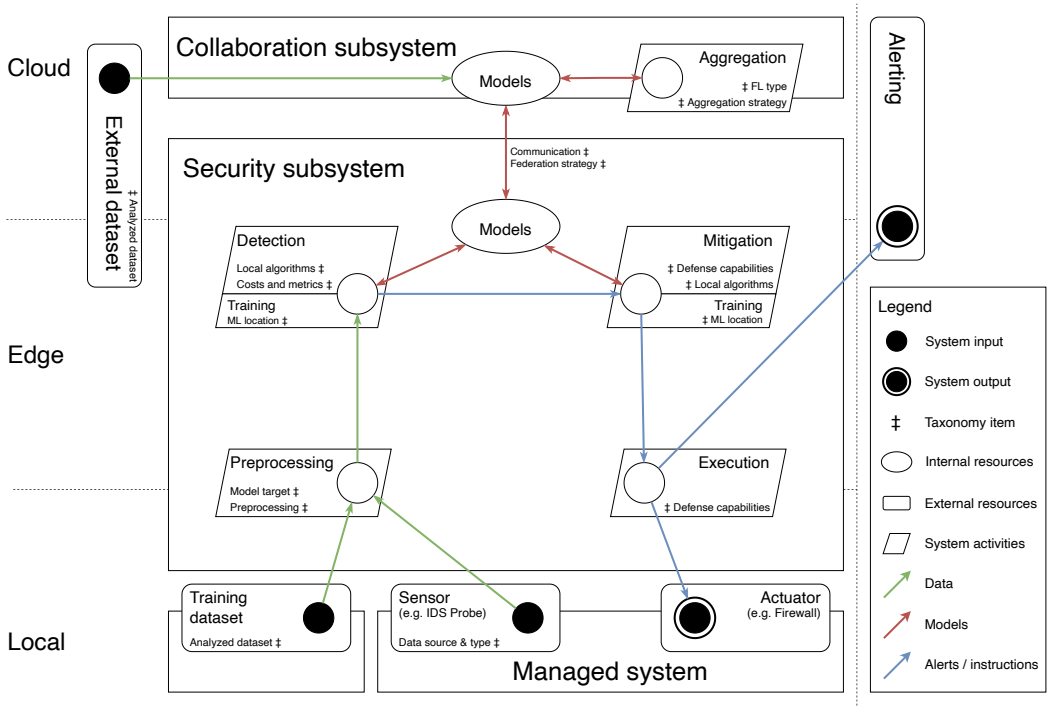


Figure 3.9 – The proposed reference architecture for FIDSs. Figure from Lavaur, Pahl, *et al.* [Lav+22b] © IEEE 2022.

— The *Managed system* represents the monitored system, *e.g.*, Information Technology (IT) network, industrial devices, or health-monitoring wearables. Collected data can either concern system or environment behavior. The former relates to information generated by the systems, *e.g.*, network traces or resource consumption. The latter refers to what the monitored system operates on, *e.g.*, health metrics for medical devices or temperature and atmospheric pressure for building management systems.

— The *Security subsystem* is the core of the architecture. It contains all the system's activities, from model training to detection and counter-measures deployment. Depending on the objectives and constraints, this subsystem can either be run locally like [PA18] or [Hei+20], on a dedicated edge-device as in [Li+20a]. In the case of centralized learning, this entire subsystem runs in the cloud. The subsystem is assumed to run a device that embeds enough computing power to perform real-time anomaly detection against ML models. It is also capable of training its own model based on collected data.

— The *Collaboration subsystem* provides the *sharing* feature of the system, essentially model aggregation. It also provides optional training from other sources, like online datasets.

This architecture has similarities with the principles of autonomic systems, as defined by IBM in 2001 [KC03], referred to as Monitor-Analyze-Plan-Execute plus Knowledge

(MAPE-K). Classic autonomic systems are local, and therefore use a database to provide *knowledge*. In FIDS, FL fills this role in the reference architecture, as the knowledge is being shared among all agents through model aggregation.

Taxonomy for FIDS

The taxonomy depicted in Figure 3.10 summaries the core components and specificities of FIDSs, as extracted from the selected works and existing related taxonomies. Correlations between the taxonomy items and the system's components can be seen in the reference architecture (Figure 3.9). It also serves as a framework for the comparisons of the selected works. Each class represents a building block, for which multiple approaches exist depending on use case and constraints.

The proposed taxonomy contains 12 classes describing the selected works that span over five main aspects:

— Two classes cover the topic of **Data**: *Data Source and Distribution* and *Preprocessing*. It defines the type of data considered and how it is distributed among clients, how it is collected, and the preprocessing strategies that are used.

— **Local operation** is represented by 3 classes: *Algorithm location*, *Local Algorithm*, and *Defense Capabilities*. It describes the detection and mitigation strategies, how models are built and trained, and where the computing resources are located.

— The **Federation** aspect is covered by 2 classes: *Federation Strategy* and *Communication*. They refer to the communication between the agents and the server, and how data sharing is organized.

— **Aggregation** is also covered by 3 classes: *FL Type*, *Aggregation Strategy*, and *Model Target*. It describes the type of FL used, how the models are merged, in accordance with the objectives of the system.

— Finally, 2 classes address the **Experimentation** topic: *Analyzed Dataset* and *Costs and Metrics*. This meta-category does not relate to the proposed solution, but to how the experiments are performed.

3.5.2 Federated Learning for Intrusion Detection

This section reviews the selected literature. Using the taxonomy as a reference, it details and compares the selected works. Table 3.2 summarizes the information and helps identify differences between the works. It gives partial answers to research questions about the components of FIDSs and how to measure their impact on performance (Questions RQ1-2.a and RQ1-2.b), while Section 3.5.2.9 replies to Question RQ1-1.b about federation techniques.

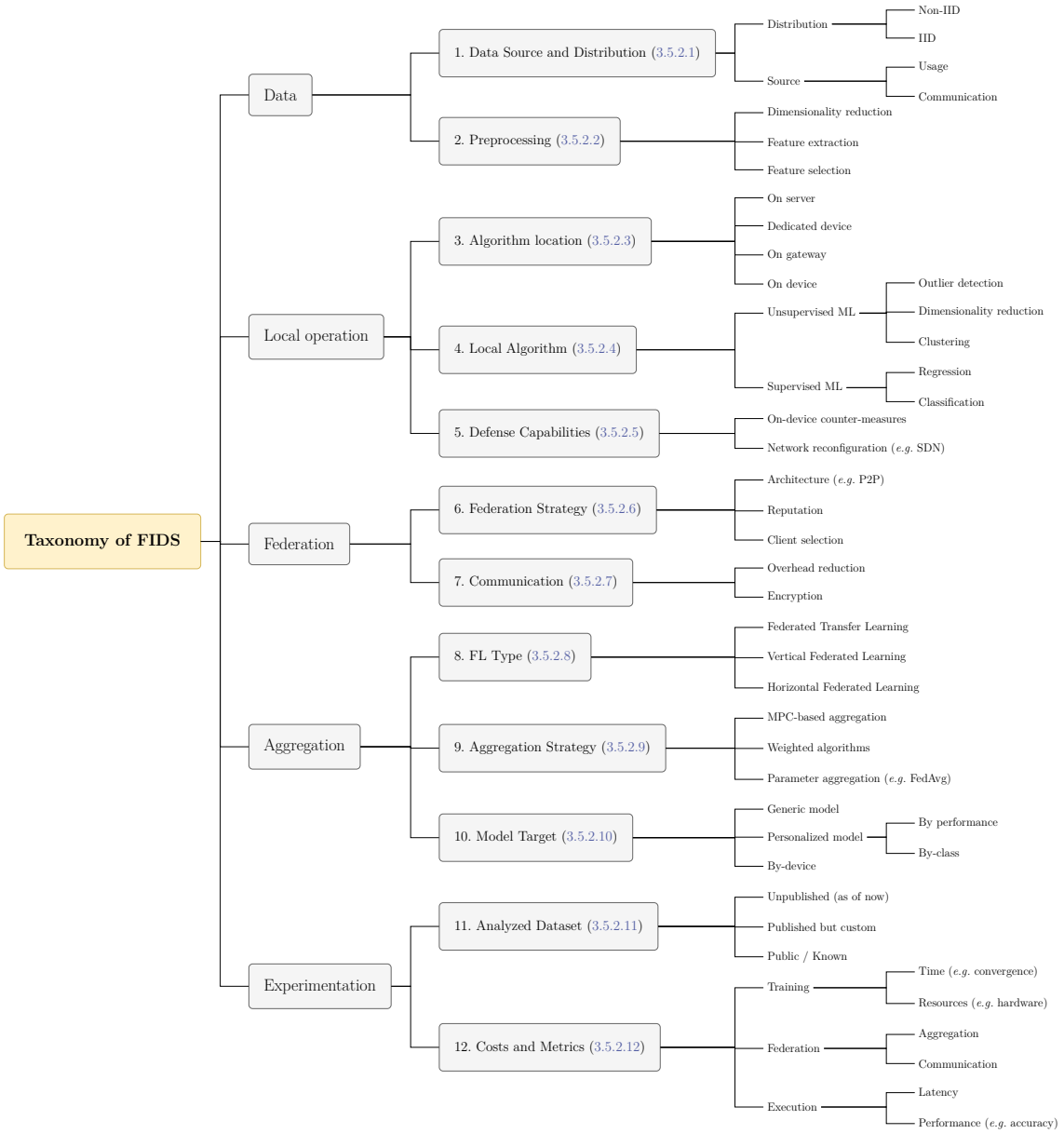


Figure 3.10 – *Proposed taxonomy for FIDS.* Figure from Lavaur, Pahl, *et al.* [Lav+22b] © IEEE 2022.

Table 3.2 – Comparative overview of selected works in the original study—approach and objectives (1/2).

	Ref	Internet of Things	Information Technologies	Cyber Physical Systems	Autonomous Vehicles	Satellite-terrestrial networks	Horizontal FL	Federated Transfer Learning	Vertical FL	Federated MTL	Federated Mimic Learning	Online learning	Supervised	Semi-supervised	Unsupervised	Personalized models	Network-based	Usage-based	Training location	Data type	Strengths
2018	Pahl and Aubet [PA18]	●	○	○	○	○	●	○	○	○	○	●	○	○	●	●	●	○	Device	Abstracted network traffic (middleware)	relatively lightweight, online, no labels
2019	Rathore, Wook Kwon, and Park [RWP19]	○	●	○	○	○	●	○	○	○	○	○	●	○	○	●	●	○	Edge-controller (SDN)	Network traffic (SDN)	offers mitigation, decentralized
2019	Schneble and Thamilarasu [ST19]	●	○	○	○	○	●	○	○	○	○	○	●	○	○	●	●	○	Gateway	IoT network traffic (TCPdump)	online, offers per-class models, no labels
2019	Nguyen, Marchal, *et al.* [Ngu+19]	○	●	○	○	○	○	○	○	●	○	○	●	○	○	●	●	○	Gateway	Encrypted network traffic (CICFlowMeter)	versatile (multi-task)
2019	Zhao, Chen, *et al.* [Zha+19]	○	○	●	○	○	●	○	○	○	○	○	●	○	○	●	○	○	●	Gateway	Healthcare sensor values	high adaptability, no labels
2019	Cetin *et al.* [Cet+19]	○	●	○	○	○	●	○	○	○	○	○	○	●	○	○	○	●	○	Gateway	Network traffic (WIFI)	–
2020	Li, Wu, *et al.* [Li+20a]	●	○	○	○	○	●	○	○	○	○	○	○	●	○	○	○	○	●	Gateway	Air conditioner sensor values	offers traceability (blockchain)
2020	Chen, Zhang, and Yeo [CZY20]	○	○	●	○	○	●	○	○	○	○	○	○	●	○	○	○	●	○	Gateway	MODBUS traffic	confidentiality (encryption)
2020	Zhang, Lu, *et al.* [Zha+20a]	○	●	○	○	○	●	○	○	○	○	○	○	●	○	○	○	●	○	Device	IoT network traffic (TCPdump)	–
2020	Fan *et al.* [Fan+20a]	○	●	○	○	○	●	○	○	○	○	○	○	○	○	●	○	●	○	Gateway	IoT network traffic (TCPdump)	no labels
2020	Rahman *et al.* [Rah+20]	○	●	○	○	○	●	○	○	○	○	○	○	●	○	○	●	●	○	Gateway	Network traffic (PCAP)	segmented (performance-based models)
2020	Sun, Ochiai, and Esaki [SOE20]	●	○	○	○	○	○	○	●	○	○	○	○	●	○	○	●	●	○	Gateway (MEC)	IoT network traffic (TCPdump, CICFlowMeter)	knowledge transfer between public and private datasets
2020	Al-Athba Al-Marri, Ciftler, and Abdallah [ACA20]	○	●	○	○	○	○	○	○	○	●	○	○	●	○	○	○	●	○	Gateway	Network traffic (TCPdump)	enhanced privacy (mimic learning)
2020	Kim, Cai, *et al.* [Kim+20]	○	●	○	○	○	●	○	○	○	○	○	○	●	○	○	○	●	○	Gateway	Network traffic (TCPdump)	–
2020	Qin, Poularakis, *et al.* [Qin+20a]	○	●	○	○	○	●	○	○	○	○	○	○	●	○	○	●	●	○	Gateway (SDN)	Network traffic (SDN)	very lightweight, line-speed classification, P4 language compatible
2020	Chen, Lv, *et al.* [Che+20]	○	●	○	○	○	●	○	○	○	○	○	○	●	○	○	○	●	○	Gateway	Network traffic (CICFlowMeter)	robust to poisoning, scalable
2020	Hei *et al.* [Hei+20]	○	●	○	○	○	●	○	○	○	○	○	●	○	●	○	○	●	○	Device	Network traffic (TCPdump)	online, offers traceability (blockchain)
2020	Li, Zhou, *et al.* [Li+20b]	○	●	○	○	●	●	○	○	○	○	○	○	●	○	○	○	●	○	Gateway	Network traffic (PCAP, CICFlowMeter, Argus)	relatively lightweight, confidentiality (encryption)
2021	Liu, Zhang, Zhang, *et al.* [Liu+21]	●	○	○	○	○	●	○	○	○	○	○	○	●	○	○	○	●	○	Gateway	IoT network traffic (TCPdump, Argus)	zero-days detection
2021	Popoola, Gui, *et al.* [Pop+21b]	○	●	○	○	○	●	○	○	○	○	○	○	●	○	○	●	●	○	Device	Network traffic (TCPdump)	relatively lightweight
2021	Qin and Kondo [QK21]	○	○	○	●	○	●	○	○	○	○	○	○	●	○	○	○	●	○	Device	Network traffic (TCPdump)	decentralized
2021	Sun, Esaki, and Ochiai [SEO21]	○	●	○	○	○	●	○	○	○	○	○	○	●	○	○	●	●	○	Gateway	Network traffic (PCAP)	segmented (performance-based models)
		Use case					FL type						Training				Approach				

3.5.2.1 Data Source and Distribution

The selected works highlight two main characteristics of the training data that impact the design of FIDSs: the origin of the data and its distribution among clients. The type of data used in the selected works is diverse, ranging from network traffic [CZY20; RWP19] to sensor values [ST19; Zha+20a]. The former is significantly more represented, probably due to the availability of public datasets like CICIDS2017 [SHG18] and UNSW-NB15 [MS15] (see Section 3.5.2.11).

Most papers [CZY20; Hei+20; Li+20a; Ngu+19; Pop+21a; Rah+20; RWP19; SOE20] use similar network features, such as source and destination, local and remote ports, TCP flags, protocol, and packet length. The authors of [Qin+20a] also target network features but at packet-level, all translated to 1D vectors: IP addresses, layer-4 protocol, ports, and IP packet length as a 120-bit input vector. Li, Wu, *et al.* [Li+20a] also explore network-related features in their use case of satellite communications. These values can be completed with preprocessing (see Section 3.5.2.2) to extract other features from the raw data. For instance, both Pahl and Aubet [PA18] and Nguyen, Marchal, *et al.* [Ngu+19] analyze the periodicity of packets, which is notably useful for volumetric attack detection. By using a middleware to classify the data, Pahl and Aubet [PA18] can train per-class models. Such models are more specialized and thus more accurate, but most communication layers do not provide such metadata. Training per-class models usually requires then a prior classification step, like in [Ngu+19]. The use of specialized models is further discussed in Section 3.5.2.10.

On the other hand, Zhang, Lu, *et al.* [Zha+20a] and Schneble and Thamilarasu [ST19] use sensor values, such as hearth rate and oxygen saturation. In this case, one does not seek to detect intrusions per se, but rather anomalies in the data that could indicate a malfunction or an attack. The observed data can be seen as a side-channel, leaking information about the actions of potential attackers. More recently, FL has been applied to Host-based Intrusion Detection Systems (HIDSs) [Guo+23], were similar considerations apply, particularly in terms of data distribution.

Finally, even when considering the same data type, use cases introduce significant differences in the available features. For instance, two systems targeting the communication between devices may encounter different protocols, services, and even communication support. In the literature, the most common use cases are (sorted by representation): Information Technology (IT), Internet of Things (IoT), Cyber-Physical System (CPS), and Autonomous Vehicles (AV). While it is unlikely that a system would target multiple use cases, discrepancies in the data distribution can exist within a single use case. Chen, Zhang, and Yeo [CZY20], and partly Hei *et al.* [Hei+20], address the topic of skewed data distribution. A non-Independent and Identically Distributed (IID) data distribution can negatively impact training performance [Yan+19]. However, most real-world scenarios

generate non-IID data, which is a major drawback of the selected works, as most of them do not address this issue.

3.5.2.2 Preprocessing

In addition to the type of data considered, the preprocessing pipeline has a significant impact on the performance of the system. Preprocessing implies the transformation of raw data into a format that can be better leveraged by ML models, either by extracting new features or by reducing the dimensionality of the data. Three main non-exclusive approaches are distinguishable in the selected works: feature extraction, feature embedding, and feature selection:

— *Feature extraction* refers to the computation of numerical characteristics after the data collection; *e.g.* Inter-Arrival Time (IAT) or number of packets per device in the context of traffic monitoring. For instance, both Nguyen, Marchal, *et al.* [Ngu+19] and Pahl and Aubet [PA18] extract periodicity features from the data. Because they only process binary features, Qin, Poularakis, *et al.* [Qin+20a] extract numerical features, and convert them to 1D vectors.

— *Feature embedding* or *dimensionality reduction* is used for algorithms that do not deal efficiently with high-dimensional vectors. We mostly use the term *embedding* when the authors use Deep Learning (DL) techniques, as it implies that the model learns the best representation of the data, such as with autoencoders [CZY20]. Other *dimensionality reduction* techniques include Principal Component Analysis (PCA), used for example by Kim, Cai, *et al.* [Kim+20].

— *Feature selection* relates to the automated selection of relevant features, before learning. For instance, Qin and Kondo [QK21] use a greedy feature-selection algorithm based on accuracy, while logistic regression can be used to eliminate features with a recursive algorithm [ACA20].

The other works [Li+20a; RWP19; ST19; Zha+20a] do not emphasize on their feature selection strategy. Moreover, some papers [Li+20a; ST19; Zha+19] use datasets that contains computed features (3.5.2.11). For experiments on live prototypes, feature computation is required.

Depending on the use case, additional features after *feature selection* or *extraction* may vary. Network analysis often relies on basic features, such as addresses and ports for source and destination, protocol, data type, packet length, and timestamp. However, these characteristics can also vary regarding their provenance: network capture [Sig99; Tav+09] or abstracted communications [PA18]. Extracted features are very common, such as inter-packet time, bytes sent per host, or bytes per packets [BG16; Cha+19]. For instance, both Nguyen, Marchal, *et al.* [Ngu+19] and Pahl and Aubet [PA18] target IoT devices, which

have a sporadic, but periodic and thus more predicable traffic. In this context, anomaly in the packet-sequence, or in the inter-arrival time might indicate an attack.

Usage-based analysis, on the other hand, is entirely dependent on the monitored device. Schneble and Thamilarasu [ST19] monitor health-related features, like arterial blood pressure or the raw ECG signals. The authors of [Zha+20a] focus on air conditioners, and therefore measure related information such as water or air temperature.

3.5.2.3 Algorithm location

The proposed taxonomy (3.10) considers three types of locations: on-device, on-gateway, and on-server. However, a large majority of the literature concerns either on-device training, or uses a dedicated device acting as a gateway. Most selected works use a dedicated device to perform the analysis, while the others assume the devices can support their own processing. Some hybrid approaches also exist, such as the multi-stage aggregation used by Liu, Zhang, Zhang, *et al.* [Liu+21], where models can be trained and aggregated at different stages of the edge–cloud continuum.

In most cases, it is the use case that dictates the model training location, as each comes with specific constraints. For instance, Zhang, Lu, *et al.* [Zha+20a] focus on a medical use case where the analyzed data solely consists of sensor measurements (Section 3.5.2.1). Connected sensors are typically lightweight devices unable to process data, so they require a gateway to be usable. Most works [ACA20; CZY20; Kim+20; Li+20a; Pop+21a; ST19; Zha+19] rely on gateways because they are more suitable for traffic analysis. It allows to capture all communications, even if the devices communicate on different supports (*e.g.*, IEEE 802.3 *vs.* IEEE 802.11). Gateway-based processing can also be motivated by the architecture of the monitored system. For instance, the authors of [Fan+20a] reuse the existing infrastructure of 5G by exploiting Mobile Edge-Computing (MEC) gateways to capture traffic and perform analysis for a 5G IoT use case. In some specific use cases, like SDN, gateways can even offer additional features that can be leveraged by FIDSs, such as packet re-routing [RWP19] or packet-level analysis [Qin+20b].

Other works [Hei+20; PA18; QK21; Rah+20] assume that end-devices are powerful enough to support their own processing. While this is generally less realistic, it can be the case for some specific use cases, like Autonomous Vehicles (AV) [Liu+21]. Indeed, such vehicles often carry consequent processing abilities for environment recognition alone, and are thus assumed to be able to perform ML training.

3.5.2.4 Local Algorithm

As discussed in the Chapter 2, one key aspects of ML for intrusion detection is its adaptability to the monitored system. Online learning refers to the ability to train a model continuously as data arrives, whereas offline learning refers to a one-shot training on a de-

fined training set. Only four of the selected works adopt online learning [Hei+20; Ngu+19; PA18; ST19]. All online work in the selection use either unsupervised or semi-supervised approaches, as continuously feeding labeled data is impracticable. Offline learning algorithms can be re-trained to adapt to new data, but this is not addressed in the selected works.

Another key difference lies in the type of algorithm used. Neural Networks (NNs), and most particularly Deep Neural Network (DNN), massively outnumber other approaches in the selected works (21 out of 22, see Table 3.3). This is coherent with the state of the art in ML for intrusion detection, as DNNs are also vastly represented in the literature. In the selection, most works rely on either Multilayer Perceptrons (MLPs) (9 out of 22) or Convolutional Neural Networks (CNNs) (4 out of 22). Recurrent Neural Networks (RNNs) are also used in 3 works, but mostly in combination with other architectures.

This sections highlights the predominance of DNNs in the selected works. These finding can be generalized to the literature published since the original study, as confirmed the more recent work of Isma'ila *et al.* [Ism+24]. Indeed, DNNs are particularly well-suited for FL:

— they are parametric models, meaning they can be aggregated using mathematical operations on their parameters (Section 3.5.2.9);

— their layers learn different levels of abstraction, enabling partial aggregation and specialized training (Section 3.5.2.10);

— their architecture can be adapted to the monitored system, as they can be trained on different types of data and for different objectives (Section 3.5.2.1).

Consequently, this choice is both relevant for intrusion detection and as a base model to be used in FL.

3.5.2.5 Defense Capabilities

Defense strategies are barely covered in the selected works, as only one paper provides actionable counter-measures. Rathore, Wook Kwon, and Park [RWP19] leverage SDN technologies, allowing the SDN controller to modify the network architecture in case of an attack. The proposed solution is tailored for Denial of Service (DoS) or flooding attacks, and therefore only needs to block the responsible traffic flow.

FIDSs could also provide remediation capabilities, providing automated resilience of a monitored system [Gho+07]. To the best of our knowledge, there is no such work in the literature. However, multiple works have been proposed to provide self-healing behaviors to information systems [Ali+18; EA10]. Other recent works have also proposed to use FL to improve event prediction in CIDSs [Nas+22]. Coupled with mitigation strategies, such systems could provide a complete solution to intrusion detection and response, enabling proactive defense against advanced attackers.

Table 3.3 – Comparative overview of selected works in the original study—algorithms and performance (2/2).

	Ref	Local Algorithm	Federation Algorithm	Accuracy	Precision	Recall	Fall-out	F-Score	K [a]	Dataset
2018	Pahl and Aubet [PA18]	BIRCH K-means	Parameter addition	0.9900	–	0.9600	0.0020	–	7	Generated
2019	Rathore, Wook Kwon, and Park [RWP19]	MLP	Early model fusion	‡ 0.9100	‡ 0.9100	‡ 0.9100	–	‡ 0.9100	15	NSL-KDD [Tav+09]
2019	Schneble and Thamilarasu [ST19]	MLP	Weight and biases average	0.9930	–	–	–	–	64	MIMIC [Joh+16]
2019	Nguyen, Marchal, *et al.* [Ngu+19]	GRU	FedAvg	–	–	0.9543	0	–	15	Generated
2019	Zhao, Chen, *et al.* [Zha+19]	FC (shared layers) → FC	Weight and biases average	∗ 0.9797	∗ 0.9634	∗ 0.9681	–	–	–	CICIDS2017 [SHG18] ISCXVPN2016 [Dra+16] ISCXTor2016 [Hab+17]
2019	Cetin *et al.* [Cet+19]	SAE	FedAvg	–	–	–	–	–	933	AWID [Kol+16]
2020	Li, Wu, *et al.* [Li+20a]	CNN-GRU → MLP	Homomorphic parameter addition	0.9920	0.9885	0.9745	–	0.9813	7	CPS dataset [MG14]
2020	Chen, Zhang, and Yeo [CZY20]	DAGMM	Parameter addition	–	0.7447	0.9803	–	‡ 0.8700	2 [b]	KDD 99 [Sig99]
2020	Zhang, Lu, *et al.* [Zha+20a]	MLP	CDW_FedAvg	∗‡ 0.8900	∗‡ 0.8600	∗‡ 0.9450	–	∗‡ 0.8500	4	Generated
2020	Fan *et al.* [Fan+20a]	CNN	Parameter aggregation	∗ 0.9100	–	∗‡ 0.9350	∗‡ 0.0020	–	4	CICIDS2017 [SHG18] NSL-KDD [Tav+09] Generated
2020	Rahman *et al.* [Rah+20]	MLP	FedAvg	∗ 0.7731	–	–	–	–	4	NSL-KDD [Tav+09]
2020	Sun, Ochiai, and Esaki [SOE20]	CNN	Parameter aggregation	∗ 0.8710	–	–	–	–	20	LAN-Security Monitoring Project [Hid18]
2020	Al-Athba Al-Marri, Ciftler, and Abdallah [ACA20]	MLP with Dropouts	FedAvg	0.9812	∗ 0.9900	∗ 0.9900	∗ 0.1320	∗ 0.9900	10	NSL-KDD [Tav+09]
2020	Kim, Cai, *et al.* [Kim+20]	MLP	FedAvg	0.9712	–	–	–	–	4	NSL-KDD [Tav+09]
2020	Qin, Poularakis, *et al.* [Qin+20a]	BNN	SignSGD	∗ 0.9640	∗ 0.9555	∗ 0.8645	–	∗ 0.9055	8	CICIDS2017 [SHG18] ISCX Botnet 2014 [Big+14]
2020	Chen, Lv, *et al.* [Che+20]	GRU-SVM	FedAGRU	∗ 0.9905	–	–	∗ 0.0108	∗ 0.9762	20	CICIDS2017 [SHG18] KDD 99 [Sig99] WSN-DS [AAA16]
2020	Hei *et al.* [Hei+20]	MLP	FedAvg	∗‡ 0.8950	∗‡ 0.9750	∗‡ 0.8775	–	∗‡ 0.9225	3	DARPA 1999 [Hai+01]
2020	Li, Zhou, *et al.* [Li+20b]	CNN	Homomorphic parameter addition	∗ 0.8100			∗ 0.1900		4	Generated
2021	Liu, Zhang, Zhang, *et al.* [Liu+21]	MLP	Parameter aggregation	‡ 0.9600	0.9400	0.9500	–	–	6	KDD 99 [Sig99]
2021	Popoola, Gui, *et al.* [Pop+21b]	MLP	FedAvg	∗ 0.9939	∗ 0.9819	∗ 0.9676	–	∗ 0.9728	5	Bot-IoT [Kor+19] N-BaIoT [Mei+18]
2021	Qin and Kondo [QK21]	ONLAD [TKM20] (ELM + AE)	FedAvg	0.7040	–	–	–	–	8	NSL-KDD [Tav+09]
2021	Sun, Esaki, and Ochiai [SEO21]	CNN	Parameter aggregation	–	–	–	–	∗ 0.8930	20	LAN-Security Monitoring Project [Hid18]
						Metrics				

∗ Value is an average of those provided by the authors.
‡ Value is read from a graph in the article, and may vary a few from the exact value.

[a] K is the highest number of client considered in the experiments.
[b] Chen, Zhang, and Yeo [CZY20] measure how one client performs, by training one other.

3.5.2.6 Federation Strategy

Another key aspect of FIDSs is how the federation is organized. This depends on the scale of the system and its architectural constraints, which are both yet again influenced by the use case. To cope with large-scale settings, massive FL applications often implement a client-selection algorithm which only train a subset of participant at each round. This reduces the computing load and bandwidth consumption at the expense of a slower convergence due to its stochastic nature—see Chapter 2. The selected works do not discuss this aspect particularly deeply, although some observed positive results from increasing the number of clients [Li+20a; Ngu+19; ST19]. Client selection can even be done dynamically [Zha+20b], even though it is not discussed in the selected works. More recent works leverage client selection, either to improve performances [Che+22], or to mitigate the risk of malicious contributions [Cun+24].

On an architectural perspective, most FL implementations follow a client-server model, where the server acts as an orchestrator distributing training tasks and model updates. This is true for most of the selected works (18 out of 22). While relatively easy to deploy, such approach has caveats, such as the necessity of trusting the central server, or the Single Point-of-Failure (SPoF) in the aggregation process [Ale+20]. To mitigate these issues, some works propose (partly) decentralized approaches, using Distributed Ledger Technologies (DLTs) to store models and updates [RWP19], or enable traceability of the training data using Merkle trees [Zha+20a]. The multi-stage aggregation proposed by Liu, Zhang, Zhang, *et al.* [Liu+21] leverages DLTs to aggregate models between Roadside Units (RSUs)—which connect vehicles to the rest of the world in the V2X paradigm. The vehicles are also able to share their models with each other, in a manner resembling gossip learning. Finally, Hei *et al.* [Hei+20] use the Hyperledger Fabric [And+18] to provide integrity and redundancy.

3.5.2.7 Communication

FL implies a significant amount of communication between the clients and the server, even though it remains more communication-efficient than distributed Gradient Descent (GD) algorithms [McM+17]. Some selected works try to reduce the communication overhead generated by their solution. Schneble and Thamilarasu [ST19] and Zhang, Lu, *et al.* [Zha+20a] compare the communication used by their system in model sharing, and compare it with the dataset size, which would require to be transferred in non-FL settings. While their results show that the relevance of FL to limit communication usage can be questioned in small datasets, its strength is undeniable with standard use cases—above 10^5 bytes according to [Zha+20a]. The communication overhead is one of the advantages of FL over centralized ML approaches.

The communication between the clients and the server can also be secured. The au-

thors of [Li+20a] and [Li+20b] use homomorphic encryption to aggregate the parameters without the server knowing the generated model. The Paillier crypto-system supports addition [Pai99], which is performed on the server, before the result is disseminated back to the clients. Each client can then decrypt the generated model, and devise the parameters by the number of participants to obtain the averaged biases and weights.

3.5.2.8 FL Type

As introduced in Chapter 2, most FL implementations use HFL, 18 out of 22. Vertical Federated Learning (VFL) is not represented in the selected works, and only found once in the updated literature selection [NDG22]. As VFL requires having the same samples but different features, it is more difficult to apply to a CIDS use case. Having the same samples would mean that the different participants monitor the same devices, just using different features, which does not follow the motivations of this work. Nevertheless, VFL might be relevant for correlation purposes in a local architecture, or between Computer Emergency Response Teams (CERTs) to share information about common threats.

On the other hand, some papers show that Federated Transfer Learning (FTL) can be used to train models in different but related contexts. For instance, a model trained on the periodicity of specific devices as in [Ngu+19; PA18] would not perform well against devices with behaviors that are too different. However, with FTL, one could quickly train a local model specific to his devices, using the knowledge acquired previously by others, as in [Fan+20a]. Another application of this concept is used by [Zha+19] with Multi-Task Learning (MTL), where a same model is trained simultaneously for multiple tasks. Like in FTL, the model is retrained after the sharing to have personalized behavior.

Al-Athba Al-Marri, Ciftler, and Abdallah [ACA20] implement Federated Mimic Learning (FML) to improve data privacy. Mimic learning is a technique that use two models and two datasets to train and share information afterward. *Teacher* model is trained on the real and sensitive data, and used to label a public dataset. *Student* model is then trained on the newly labeled public dataset, and shared with other participants after that.

3.5.2.9 Aggregation Strategy

The aggregation strategy is at the core of FL, as it was the original contribution [McM+17] separating it from distributed GD algorithms. More specifically, the base principle of training over multiple epochs locally and aggregating the models (*i.e.*, `FedAvg`) is one of the key components of FL. This approach is the base of most implementations going forward. Naturally, it is also the most represented aggregation algorithm in the selected works, as 6 out of 22 [ACA20; Kim+20; Ngu+19; Pop+21a; QK21; Rah+20] use it directly. Others leverage alternative weighting mechanisms, where `FedAvg` weights models based on the number of samples they have been trained on.

Zhang, Lu, *et al.* [Zha+20a] propose to weight the aggregation according to the centroid distance between the positive and negative classes of the client. They claim it reduces the impact of heterogeneity in the data distribution. Other articles average the parameters of the uploaded models [CZY20; ST19], while not mentioning `FedAvg` explicitly. The aggregation weights also represent an opportunity to balance the clients contributions. This is widely used in the FL literature, whether it is based on the number of samples [McM+17], on the participants' obtained reputation [Wan+22; WK21], or based on model-quality metrics [Den+21].

Because they use Binarized Neural Networks (BNNs) with only binary values, the authors of [Qin+20a] cannot simply average the model parameters. While the last layer of the BNN could be converted to numerical values to be aggregated more easily, the authors prefer the binary approach SignSGD [Ber+18]. This aggregation algorithm relies on majority voting to estimate the best weights for the layers. While their system performs well, the authors point out that updates that do not change the sign of the weights represent a waste of resources, since only two values are possible, $+1$ or -1.

Further, we also considered works in the literature that push the boundaries of the aggregation process, even if it does not suit the formal definition of FL. For instance, Rathore, Wook Kwon, and Park [RWP19] use early model fusion, a technique that concatenates the feature vectors of the models to learn the best feature representation. Pahl and Aubet [PA18] use Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) clusters, which have the particularity of being easily aggregated by simply adding the features of multiple clusters together. Timestamps are also saved to detect the *staleness* of the clusters.

Finally, multiple aggregations can be performed in a single system, whether it is hierarchically [Liu+21] to reduce the communication overhead, or in different clusters in parallel [SEO21; SOE20] to reduce the heterogeneity inside communities. The recent literature contains more works leveraging clustered FL to build more homogeneous communities [Cai+22; Sha+24]. Clustering will be extensively leveraged in Chapter 6 to detect malicious contributions in heterogeneous settings.

3.5.2.10 Model Target

The existing literature on FIDS highlights how building a generic efficient model is difficult. Nguyen, Marchal, *et al.* [Ngu+19] stress that anomaly detection systems suffer from lower performance when monitoring multiple behaviors at the same time. This especially impacts the false positive rate and sensitivity in their experiments. To address this issue, they propose to use an autonomous classification system [Mar+19] to categorize devices first, and then train per-class models. However, this classification problem is not specific to intrusion detection, and standards have been proposed for devices to advertise such information. Manufacturer Usage Description (MUD) [LDR19] for instance allows

devices to signal to the network what type of functionalities and authorizations that they require for operating properly. While they do not rely on an existing standard, Pahl and Aubet [PA18] use a middleware providing similar feature by communicating predefined classes attached with each device's requests.

Fan *et al.* [Fan+20a] leverage FTL to enable model specialization. Each client trains a personalized version of the global model using Transfer Learning (TL). This allows to train models accurately on the singularities of each network, while improving the overall performance of the system. Another approach is to use MTL to train a single model on multiple tasks. Zhao, Chen, *et al.* [Zha+19] train a unique model for anomaly detection, Virtual Private Network (VPN) traffic classification, and TOR traffic recognition. Qin and Kondo [QK21] propose another way of building different more specific models, by training models depending on the feature set used by the local device. They emit the hypothesis of building models per attack: devices could train a model for *DoS* attacks, others for *Probes*. The other works considered in this survey use a global model for their detection [ACA20; Che+20; CZY20; Hei+20; Kim+20; Li+20a; Pop+21a; Qin+20a; Rah+20; ST19; Zha+20a], regardless of the data type or detection method.

3.5.2.11 Analyzed Dataset

The literature on IDS has produced various datasets over the years, which are used to evaluate the performance of the proposed systems. Common datasets include KDD'99 [Sig99] and its fixed version NSL-KDD [Tav+09], UNSW-NB15 [MS15], and CICIDS2017 [SHG18]. We will not describe these datasets here, as Chapter 2 already provides an overview of the most common datasets used in intrusion detection. However, it is worth stressing again that these datasets are often criticized, either for their age, their lack of realism, or the different biases they contain. For instance, NSL-KDD fixes multiple issues of the original KDD'99 dataset, such as removing redundant and duplicated records. Likewise, recent works [ERJ21; Lan+22] have demonstrated issues in the CICIDS2017 dataset, *e.g.*, duplicated records, ineffective attacks, and misordered packets.

The initial selection highlights a consequent representation of said datasets: out of the 22 selected works, 3 use KDD'99, 6 use NSL-KDD, and 4 use CIC-IDS2017. There are still some overlaps between the datasets, as some works [Che+20; Fan+20a; Qin+20b; Zha+19] test their approach on multiple datasets (but only one at a time). The other works generally use domain-specific datasets, such as the MIMIC-III dataset [Joh+16] for health-related data [ST19], or the CPS dataset of Morris and Gao [MG14] for FIDSs in industrial settings [Li+20a].

However, this selection highlights a massive drawback of the literature: biased assumptions on the data distribution. Except for Sun, Ochiai, and Esaki who used a dataset collecting data from effectively distributed sources, all selected work use a single dataset and distribute it among the clients. Even worse, most of them either do not mention

data-distribution at all, or assume it is IID. As discussed in Section 3.5.2.1, this is a major drawback, as most real-world scenarios generate non-IID data. This makes most experiments unrealistic, as the clients train their models on data generated on the same network topology, with the same devices, and the same behaviors, even considering Non Independent and Identically Distributed (NIID) settings.

Just after we stopped data collection for the original study, Sarhan, Layeghy, and Portmann [SLP22] proposed a standardized feature set for intrusion detection (see Chapter 2), and converted four known IDS datasets to this format: UNSW-NB15 [MS15], Bot-IoT [Kor+19], ToN_IoT [Mou+20], and CSE-CIC-IDS2018 [SHG18]. The uniform feature set across datasets allows FL-based approaches to evaluate their performance on independently generated datasets [dCar+23; Pop+21b], closing the gap towards more realistic experiments. In the context of cross-silo FL, each dataset can act as one organization's collected data, which is done by **de_carvalho_bertoli_generalizing_2023**. Finally, we present in Chapter 7 our work on topology generation, which will provide building blocks for generating more realistic distributed datasets.

3.5.2.12 Costs and Metrics

We can divide the metrics used in the selected works into three categories that follow the life cycle of the system: training, federation, and execution. Training-related metrics measure the behavior of the model during the training phase. Federation-related assess the costs and benefits of the FL approach, while execution-related metrics measure the performance of the system in real-time, notably during the detection or classification phase.

Training This includes typical metrics like accuracy and loss used during the training phase, as well as resource-related metrics. They can be used to measure the convergence time of the model, often characterized as obtaining an accuracy above a defined threshold (*e.g.* 90% in [CZY20]), or with the percentage of loss improvement between two epochs (*e.g.* 0.01 in [Kim+20]). Training time also serve as a comparison between approaches [ST19], even though it depends a lot on the underlying hardware architecture. Finally, it can be used as a metric to select other hyper-parameters, such as the number of epochs in [Liu+21]. Algorithm complexity and resource consumption are also relevant metrics to measure local training costs. Constrained use cases like IoT require complex algorithm to run on resource-limited devices. In [PA18], the authors also study complexity to choose BIRCH clusters instead of K-means, as updating the former is easier—$\mathcal{O}(d)$ *vs.* $\mathcal{O}(n * d)$, where d is the dataset size. Hardware-related resources are used by [RWP19; Zha+19], mostly to emphasize differences between their approach and another, often more standard one. These resources often include CPU, disk and memory usage, as well as energy consumption. However, evaluating hardware-related metrics requires experiments to

be implemented using the same hardware and software stacks. Hardware- and energy-based metrics are especially relevant in constrained scenarios [Ngu+19; ST19], whereas training time is relevant for most use case, while not a priority. When these measures are collected on reference hardware, it can also be used to evaluate the feasibility of the approach, as in [Ngu+19], if the hardware matches the deployment constraints of the study.

Federation Federation-related metrics are heavily tied to the communication between clients, or with a server. The communication overhead is a core metric of FIDSs, as high bandwidth consumption is a drawback of CIDSs (see **??**), especially in constrained environments [Qin+20a]. The overhead is often measured in bytes, either per packets [PA18], or for the total of all communications [ST19; Zha+20a]. Metrics must be adapted to the specificities of each solution, for instance when adding a feature. Consequently, Zhang, Lu, *et al.* [Zha+20a] add specific metrics in their evaluation to measure the impact of using the blockchain, like the time of the *blockchain encoring process*. Some works [ACA20; CZY20; Fan+20a; Li+20a; Pop+21a; Rah+20; RWP19; SOE20], on the other hand, do not cover federation-related metrics in their evaluation, which is questionable as it is a critical part of FIDSs.

Execution Finally, execution-related metrics are mostly focused on performance, and often come from the ML community. As shown in Table 3.3, *accuracy* is used by almost all reviewed works, followed by *precision* and *recall*. More generally, all metrics issued from the confusion matrix can be used, but the literature emphasizes on metrics that focus on the detection of anomalies, like *recall* and *precision*, or the *F1-score* which combines the two. Researchers often use these metrics to compare their results with related works. Other execution metrics like execution time are considered, as it can be critical for intrusion detection tasks. Latency allows a comparison between different architectures, especially *centralized*, *distributed*, and *decentralized* [RWP19]. Latency is also relevant for highly constrained setups, as in [Qin+20a]. As pointed out in Section 3.5.2.3, *ML location* can have an impact on data collection, but also on detection latency, if data need to travel over network to be analyzed. Execution metrics are only relevant when comparing works that share implementation. Such comparison is often performed by reimplementing a selection of related works. They can also be used to highlight differences between approaches, like between *local*, *federated*, and *ideal* models [Li+20a; RWP19].

3.6 Discussion

This section first discusses the limitations of this study, notably the number of selected papers and the methodology used. We then answer Question RQ1-3.c by identifying the

open issues and according research directions, and associate them with recent publications.

3.6.1 Limitations of this Study

The original review reviewed 22 technical papers about FIDSs, selected using SLR methodology. This ensures that the selected papers are representative of existing works in this field. Other surveys in similar but broader fields worked with bigger quantities of papers; 231 in [Lo+21] about FL, or 95 in [dCos+19] for ML-based IDSs. Therefore, all conclusion extracted from the selected works must be put in perspective of the number of analyzed papers.

Furthermore, SLR methodology guarantees the exhaustive aspect of the selection. However, relevant papers may have been missed; especially, edge–use-cases and unusual wording can exclude papers from the selection process. We expect the steps presented in Section 3.2 to mitigate this risk, notably snowballing.

Moreover, the selected metrics give insight on the quality of the predictions, and more importantly the comparison between FIDS and local detection, when provided. As the selected works target different use cases with different objectives, a performance metric-based comparison is less relevant. Using the same datasets, hardware and network configuration, and coding frameworks, a thorough reimplementation of the reviewed papers could provide significant contributions.

Finally, the selected papers are from 2016 to 2021, and the field of FIDS has been evolving rapidly. As noted in Section 3.2.3, a significant number of papers were published since the end of data collection of the original review, including new literature reviews (see Section 3.3). Yet, the conclusions of this study have provided a solid foundation for the other contributions of this thesis and others in the literature. Further, most trends and research directions that we identified are still relevant in 2024.

3.6.2 Open Issues and Future Directions

As FL is becoming more mature, new research tend focus either on side-aspects like security and privacy [Bon+17; Don+20; FYB20; Mot+21b; Ngu+20] or on its application to a specific use case, as do the works selected in this survey. This section reviews open questions identified by literature, and the proposed according research directions. Additionally, for each identified open issue, we provide relevant publications that have been published since the original review to complement the discussion. Some of these issues depend on works from other related fields, such as ML for performance or FL for scalability. However, the specificities of FIDSs require dedicated research. Especially, the topics of security, trust, and heterogeneity are critical for a collaborative security use case.

Performance Like any detection system, FIDSs are looking for an absolute performance: a system with a perfect classification score, producing no false positives or negatives. To this end, several research directions have been identified by the selected works, such as the use of Generative Adversarial Networkss (GANs) [ST19] or the improvement of feature selection as input to the model [SEO21]. More generally, there is a need for a better understanding of the impact of hyper- and meta-parameters on the performance of the system. This is especially true for FL, where the aggregation process can be seen as an optimization problem in itself [CK21]; a problem for which the right parameters need to be inferred. Both GANs [Jin+24] and meta-learning for local data-sampling [HDH23] have been reviewed as potential solutions to this problem.

Adaptability Constrained environments like low-bandwidth networks, or low-powered devices, may also impact the ability of FL to provide detection in a timely fashion (Chapter 2). Since the security of constrained devices is a growing concern, the selected works identify relevant research directions in this area, such as implementing compression algorithms [Fan+20a] or globally reducing the number of computation rounds [Rah+20]. Moreover, as time goes by, the training data can be easily become outdated. Updating strategies need to be studied to provide accurate results as time goes [Fan+20a], and adapt to changes in the traffic behavior [QK21]. This topic has been especially tackled in FIDSs using incremental learning [Jin+23].

Scalability Distributed systems such as FL are often used to cope with resource limitations, especially in terms of computation and bandwidth. However, as pointed out by several selected works, FL faces limitations when dealing with too many clients [Fan+20a; RWP19]. Therefore, FIDSs require further research regarding client selection: performance-, time-, or reputation-based [Cun+24]. Moreover, in massively distributed federations, the aggregation process can become a bottleneck. In such settings, researchers and practitioners might consider using hierarchical aggregation or even complete decentralization of the system. A few decentralized FIDSs approaches have been proposed since [Fri+23].

Heterogeneity and Transferability The approaches presented in the initial review mostly consider that all local models share the same architecture and hyper-parameters, use data from the same domain, and that all clients possess similar resources. These limitations hurdle convergence, and more generally make current FIDSs less versatile and transferable. Hence, open issues include allowing the federation of cross-domain clients [Li+20a]. As pointed out in Section 3.5.2.1, the features selected for model training have to be applicable to multiple environments. Transfer learning [SA21; She+20] and its federated variant FTL [Che+20; Fan+20a] have been applied to similar domains in the past, and might also represent a favorable direction for future research in terms of adapt-

ability. Since the submission of this study, multiple papers [Che+22; Kho+21; OWN21] have been published in this direction.

Security and Privacy The broad attack surface of FL directly applies to FIDSs, raising concerns about poisoning, inference, or model extraction. The selected works already address some of these issues by leveraging homomorphic encryption to secure the aggregation process [Li+20a; Li+20b]. Others identify this aspect as potential future works [Che+20], with countermeasures like MPC or Differential Privacy (DP). Furthermore, as ML, and especially DL, lacks explainability, the content of a model is difficult to infer. It complicates the detection of poisoning attacks, as it is hard to distinguish between a model that has been poisoned and one that has been trained on a different dataset. Poisoning has recieved a lot of attention in the literature of FL [FYB20; Ngu+20; Ngu+22], and some works have been published in the context of intrusion detection too [Mer+23; Yan+23].

Trust and Reputation Following the same line of thought, the trustworthiness of the participants is a critical aspect of FIDSs. Malicious participants can indeed impact the model and the detection process. More generally, the quality of the participants' contributions must be controlled to ensure the quality of the aggregated model. Zhang, Lu, *et al.* [Zha+20a] identify assessing the trustworthiness of the participants as a future research direction. Inspiration should be taken from the state of the art of collaboration systems and information-sharing platforms, which address problems such as trust or reputations [SSF16; Wag+19], which are relevant for FIDS. Since the submission of this study, works have been published on the topic of trust via client selection [Cun+24]. Chapter 6 presents a solution to this issue, by providing a trust-based model weighting mechanisms to ensure the quality of the aggregated models [Léo+24].

Self-defense and self-healing As highlighted in Section 3.5.2.5, current research on FIDS is focused on intrusion detection and attack classification. Mitigation is barely represented in the literature [RWP19]. However, technologies like SDN offer quick mitigation capabilities, and recent works study the effectiveness of such defense mechanisms [BG17; SB20]. New emerging applications like self-defense and self-healing systems could benefit from FIDS and other FL-based technologies. A handful of works have been published on the topic of attack mitigation and reaction [dCal+23; Pan+22; PG22], corroborating our survey's findings.

Evaluation Finally, the topic of evaluation raise two major issues in the selected works. First, reproducibility is a major concern, as few are the works that provide the code or the datasets used for the experiments. This is a common issue in the field of ML, which has long been criticized for its lack of reproducibility [Arp+22; Baj+17]. The same issue

is present in the field of FIDSs, as most of the selected works do not provide enough information to reproduce the experiments. Some do not even disclose the datasets they used, such as Nguyen, Marchal, *et al.* for DÏoT [Ngu+19]. Further, existing public datasets are not representative of FIDSs deployment environments. Indeed, they are often datasets produced for traditional ML-based IDSs, but split for federated purposes. This makes most of the literature biased, as all samples are related to the same original event. Recent works have tried to partially address this issue by providing standardized datasets [SLP22] or dedicated ones [Fer+22], although the problem remains unsolved.

3.7 Conclusion and takeaways

FL comes solves two main challenges: (1) it breaks isolated architectures by allowing learning over distributed data without compromising privacy; and (2) it speeds up training and reduces communications compared to existing distributed learning approaches, and even more so when compared to centralized learning. Applied on intrusion detection, FL allows to leverage the knowledge of multiple actors to improve the detection of attacks, while preserving the privacy of each organization's data. This is particularly relevant to fit with the injunctions of security agencies and regulations, which call for collaboration and intelligence sharing, while also demanding strong privacy requirements. Based on the literature reviewed, we can define FIDSs as follows:

Definition 3.1: Federated Intrusion Detection Systems (FIDSs)

Distributed IDSs with privacy-preserving federated knowledge. FIDSs leverage FL or similar distributed learning techniques [1] to share and aggregate the models trained locally with other members of the federation. Federations can be closed (*i.e.*, all participants are identified and trusted) or open (*i.e.*, participants can join and leave the federation at any time). Depending on the ML model used locally, training can happen offline on labelled data, online, or with a combination of both.

This review highlighted eight main challenges that need to be addressed to build efficient and secure FIDSs, ranging from pure performance to scalability and mitigation mechanisms. In the following chapters, we will especially focus on three of these challenges: (i) *FIDSs in Heterogeneous Environments*; (ii) *Malicious Contributions and Trust*;

1. We take some liberty in this definition by not imposing the use of FL as a requirement. While it is definitely the most popular approach and the motivation behind this study, other privacy-preserving distributed learning techniques have been used in the literature [PA18; RWP19]. Further, the formal definition of FL is still debated (see Chapter 2), as the term is often used to describe a broader set of techniques.

and (iii) *Evaluation and Dataset Representativity*. More specifically, we will address the following points:

— We address *Heterogeneity* and *Dataset Representativity* in Chapter 7, where we propose a novel approach to generate network topologies that are both realistic and heterogeneous.

— Chapter 5 reviews the impact of *Malicious Contributions* over FIDSs in IID settings, with an emphasis on reproducibility.

— We propose in Chapter 6 a novel approach to mitigate such effects in *heterogeneous* environments, leveraging *reputation* systems to assess the quality of the participants' contributions and their *trustworthiness*.

Before diving into these challenges, we will first present in Chapter 4 an application of FIDSs using toy examples on public datasets. This will allow us to visualize the potential of FIDSs, but most importantly the impact of the challenges we identified above.

Chapter 4

Performance and Limitations of FIDSs

4.1 Introduction

In the previous chapters, we have discussed the perspectives offered by applying Federated Learning (FL) to Intrusion Detection Systems (IDSs), notably in terms of collaboration. Based on the insights gained from the literature, it is now clear that FL can be used to train a global model over the distributed data of a federation of organizations. It even seems that FL could be used to share attack knowledge, still without sharing participants' local data.

In this chapter, we present critical examples showing the challenges that arise when applying FL to Collaborative Intrusion Detection Systems (CIDSs). We start by laying out in Section 4.2 the practical use case that will be used throughout the rest of the manuscript. Then, we highlight some limitations of FL in the context of CIDSs in Section 4.3, based on our demonstration paper published at ICDCS 2024 [LBA24a].

Contributions of this chapter

- A practical use case for FL in the context of CIDSs involving multiple organizations.
- A exhibition of the limitations of FL in the context of CIDSs, notably in terms of data heterogeneity and susceptibility to poisoning attacks.

4.2 A Practical Use Case for FIDSs

We consider a typical FL scenario where a central server S is tasked with aggregating the model updates w_k^r of a set of participants $P = \{p_k | k \in [\![1, n]\!]\}$ at each round r. The participants p_k are entities that oversee an organization's network, which makes them highly available and interested. This can be described as a Cross-Silo Federated Learning (CS-FL) scenario, *i.e.*, fewer participants with consequent amounts of data and significant computing capabilities. Because of the lower scale of the federation and the

assumed interest of the different parties, we set the fraction C of participants that are selected at each round to 1.

For the sake of simplicity, we consider that all participants share the same model architecture and extract the same features from the network traffic. This is not unrealistic, as common formats and protocols are used in the industry for this purpose, such as Cisco's NetFlow format [Cla04] for network flows. Further, this description can fit multiple scenarios, such as organizations deploying the same probe in their network as part of a standardization effort, or a service provider offering a gray-box product to multiple organizations. Although the features are assumed to be identical across participants, the distribution of the data can vary considerably, as each organization has its own network configuration and security policies [ZLK10].

We also consider that participants have access to labeled data, which is a common assumption in the literature. Although labeling data can be costly, it is a more reasonable assumption in CS-FL scenarios, where participants are more likely to have the human and financial resources to label data. Therefore, each participant possesses a local dataset $d_k = (\mathcal{X}_k, \mathcal{Y}_k)$ that is not shared with the others. Because of the differences between organizations, the distribution of each local dataset d_k can vary considerably, independently of the associated labels. Indeed, the same network behavior (say Peer-to-Peer (P2P) file sharing) might be considered normal in an organization (*e.g.*, a media company) but flagged as suspicious or outright malicious in another (*e.g.*, a financial institution). However, the CIDS use case implies that similarities can exist between participants, for instance between organizations operating in the same sector or having similar network infrastructure. This particular setting can be described as *practical* Non Independent and Identically Distributed (NIID), as opposed to the *pathological* NIID settings, where all participants have unique and highly different data-distributions [Hua+21]. This is the most common setting in Federated Intrusion Detection Systems (FIDSs), as it serves the goal of improving behavior characterization, and having access to knowledge that cannot be inferred with only local data.

4.2.1 Dataset selection

Since we consider that all organizations share the same model architecture, we need multiple independently-generated datasets that share the same feature set. Fortunately, Sarhan, Layeghy, and Portmann [SLP22] have proposed a standard feature set for IDS datasets, based on NetFlow v9 (see Section 2.2.2). Namely, we used the modified versions of the following datasets:

— UNSW-NB15 [MS15] is produced using the IXIA PerfectStorm tool on the Cyber Range Lab of UNSW Canberra. The traffic is a hybrid set of real modern normal activities and synthetic contemporary attack behaviors, grouped in 9 attack classes.

— Bot-IoT [Kor+19] is another dataset generated at USNW, using a realistic smart home environment setup, completed by IoT devices. It focuses on the detection of IoT botnet attacks, the DoS and DDoS classes being the most represented. This dataset is highly unbalanced, as the majority of the traffic is malicious.

— ToN_IoT [Mou+20] is yet another dataset generated by the same team, containing IoT/IIoT telemetry data, network traffic, as well as system logs. The network dataset contains 9 attack classes, including Ransomware, Scanning, and XSS.

— CSE-CIC-IDS2018 [SHG18] is a dataset generated by the Canadian Institute for Cybersecurity in collaboration with the Communications Security Establishment (CSE). The traffic is collected on a large-scale infrastructure deployed on AWS. It contains 14 attack labels, grouped in 6 attack classes.

In most of the experiments presented in this manuscript, We use the "sampled" version (1,000,000 data points per dataset) provided by the same team [LP22] We remove the port and IP addresses for both source and destination, as they are rather a representation of the network topology and device configurations than of traffic patterns [dCar+23]. We then use one-hot encoding (see Section 2.2.1) on the categorical features (both in the sample and labels), and apply min-max normalization to give all features the same importance in model training. This pre-processing step produces 39 features for each sample.

4.3 Exhibiting the Limits of FIDSs

This demonstration spans over four specific scenarios, each highlighting a specific aspect of the considered challenges. The first three (Sections 4.3.2 to 4.3.4) target different heterogeneity scenarios, ranging from homogeneous dataset partitioning to completely independent data sources. The last scenario (Section 4.3.5) focuses on poisoning attacks against FL, where malicious participants try to degrade the performance of the global model.

4.3.1 Setup

To evaluate the performance of FL in the context of CIDSs, and especially evaluate the feasibility of the scenario presented in Section 4.2, we need datasets that are representative of the traffic that can be observed in real-world networks. Consequently, we use the datasets mentioned in Section 4.2 with the NF-V2 format, which allows us to use the same model architecture for all participants.

To generate the different scenarios, we build an evaluation framework for FL called Eiffel [1] [LBA24a], which relies on Flower [Beu+20], a modular FL framework. Eiffel is a

1. Available at: https://github.com/phdcybersec/eiffel

Table 4.1 – Parameters used for all scenarios.

Parameter	Notation	Value
Federated Learning		
Number of rounds	R	10
Local epochs per round	$\mathcal{E}$	10
Number of clients	K	4
Local Training		
Neurons of the (2) hidden layers		128
Activation function (hidden layers)		ReLU
Activation function (output layer)		Softmax
Batch size	β	512
Learning rate	η	0.001
Datasets		
Number of features		39
Number of samples		100,000

Python library that provides a set of tools to automate the evaluation of FL algorithms, such as instantiating various types of data distribution, local models, and aggregation strategies. It further provides multiple label-flipping attacks, and automates metric collection and plotting to quickly evaluate the impact of each parameter.

To assess the impact of a scenario on the federation, we evaluate the global model on each participant's test set and collect different performance metrics. The results are averaged over the different participants to obtain the global model's performance. We select the F1-score as the main metric for its focus on positive samples, but the same methodology can be applied to other metrics. To assess the performance of a model trained only on local data, we define a `FedNoAgg` strategy, where local models are kept by participants at the end of each round. Therefore, models are trained during $\mathcal{E} \times R$ local epochs, where R is the number of rounds and $\mathcal{E}$ is the number of local epochs per round, instructed by the server. Table 4.1 summarizes the parameters used for all scenarios, with the notations defined in Section 2.4.

4.3.1.1 Data Partitioning in IDS contexts

Pathological-NIID partitioning is rarely seen in IDS binary-classification tasks, as they typically require both benign and malicious training data. Therefore, a common NIID partitioning scheme is: 1. *pathological*-NIID of the attack classes, *e.g.* one or two class per client; and 2. Independent and Identically Distributed (IID) benign samples. Campos *et al.* [Cam+22] also review other partitioning settings based on the ability to separate data by client IP in public datasets. They also artificially build balanced IID partitions

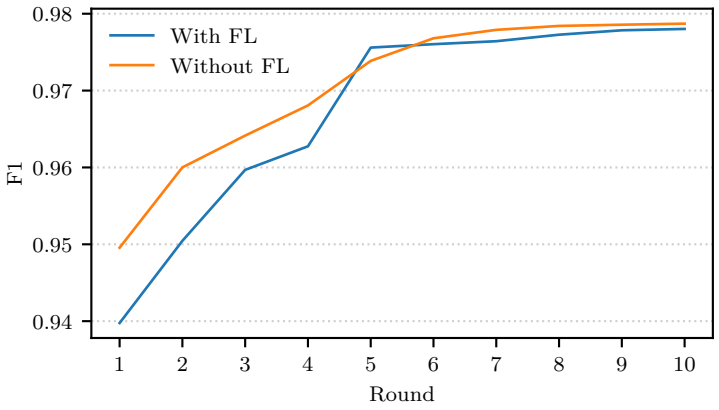


Figure 4.1 – Global model performance in IID.

by dropping attack samples until a specific Shannon entropy threshold window is reached for the local distribution. This approach is however more suited for cross-device use cases, as each client receives the data from one device only. Overall, NIID data for a cross-silo Network-based Intrusion Detection System (NIDS) context is typically one of:

(a) distributing a dataset among clients, before removing samples from n attack classes from each client; or

(b) distributing the benign data among clients, before giving samples from n attack classes to each client, with or without class overlap.

In this chapter, we use two approaches to generate NIID data:

(a) a *practical* NIID partitioning, where each client loses two attack classes; and

(b) a more *realistic* NIID setting, where each client has a different dataset.

4.3.2 Scenario 1: IID Data

The first scenario is the simplest one, where the data is partitioned in IID settings. Each participant receives $\frac{N}{C}$ samples, after shuffling the dataset. Figure 4.1 presents the results of this scenario based on the global model's F1-score. There are virtually no differences between the `FedNoAgg` and `FedAvg` strategies, since each participant has enough samples of each class to train a suitable local model. Therefore, there are little benefit to using FL in this scenario.

However, this configuration is often found in the literature to evaluate CIDSs based on FL, such as in [Aou+22]. While this experiment illustrates the lack of performance gains on IID data, larger-scale setups configurations might benefit from FL. In fact, selecting only a subset of the available participants could obtain similar results while reducing the local computing costs for participants. This setup is thus more akin to a distributed learning approach, where the server is only used to coordinate the training process.

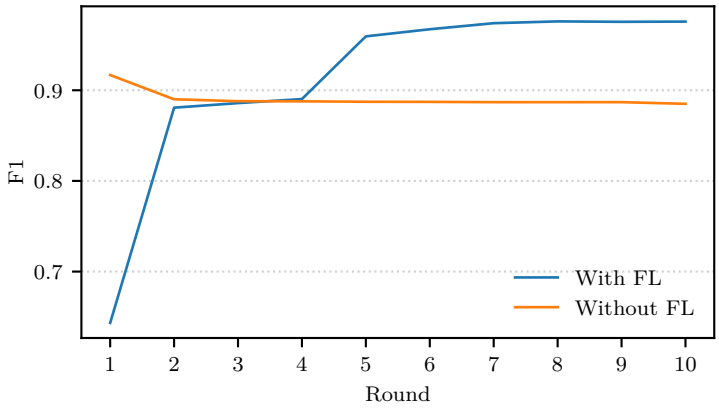


Figure 4.2 – Global model performance in NIID (same source).

4.3.3 Scenario 2: NIID Data from the Same Source

The second scenario highlights the knowledge-sharing capabilities of FL, as it can transfer characteristics of the data distribution between participants. To illustrate this, after partitioning the data as in Section 4.3.2, we randomly drop two classes from each participant's train set. This results in a NIID data distribution among participant, where each one has a different subset of classes. Figure 4.2 displays the results of this scenario, where `FedAvg` performs significantly better overall than having clients train locally. However, the F1-score hides the fact that some participants can miss entire attack classes in the test set, rather than it being a global model issue.

Specifically, since clients have different subsets of classes, they might be unable to detect some intrusions that are not present in their training data. For example, Table 4.2 displays the Detection Rate (DR) of the first client (`client_0`) in our setup for each attack class, both in local and federated training, along with the number of samples of each class. `client_0` has no samples of the `Infiltration` and `DoS` classes, and therefore cannot detect them, *i.e.* its DR is either 0 or very low. However, the global model is able to detect these classes, as other clients have samples of these classes in their training set. We also see a slight decrease in performance for the other classes (*e.g.*, 99.91 instead of 100 for `DDoS`) due to the aggregation process. Note that the `Infiltration` being only detected at 20.11% by the global model is the expected behavior on this dataset, as it is particularly difficult to detect (see the baseline results in Chapter 5 for more details).

These results indicate that FL can effectively share knowledge between participants, allowing them to detect attacks that are not present in their local training data. This is a key feature of FL in the context of intrusion detection.

Table 4.2 – Detection rate (DR) of `client_0` in NIID settings. Rows where knowledge-sharing is visible are highlighted in gray.

Attack class	Samples	DR (local)	DR (federated)
DDoS	176107	100	99.91
DoS	0	2.43	98.57
Bot	1513	100	99.94
Brute force	1299	99.77	99.55
Infiltration	0	0	20.11
Injection	3	100	100

4.3.4 Scenario 3: NIID Data from Different Sources

While we highlight in Section 4.3.3 that FL can benefit from having different datasets per client, to the point where it can share knowledge between participants, the third scenario illustrates the limits of this assumption. CIDS experiments in the literature often evaluate their approach with a scenario close to the ones presented in Sections 4.3.2 and 4.3.3, where one dataset is partitioned among participants. However, in practice, participants will likely collect data from different networks, and therefore have different data distributions.

In this third scenario, we test `FedAvg` in this configuration, with each participant having a different dataset. Thanks to the standardized feature set (see Section 4.3.1), we can use the same model architecture for all participants, which is a requirement for `FedAvg`. The class overlap between datasets is also not an issue in this use case, as we focus on binary-classification, which implies that all participants have benign and malicious samples.

The results presented in Figure 4.3 confirm great performances overall when participants are trained locally. However, the global model's performance is highly impacted by the heterogeneity of the data distributions. This is likely due to the fact that all participants converge to local minima that are too different from each other, and therefore the aggregation do not result in a suitable model for all participants. Other approaches than `FedAvg` have been proposed to address this issue in IDS context, as the one by Popoola, Gui, *et al.* [Pop+21b] for instance, who use `Fed+` [Kun+22] as the aggregation strategy and present promising results in a similar scenario.

4.3.5 Scenario 4: Poisoning Attacks

With the first three scenarios, we have highlighted how the heterogeneity between participants can impact the performance of FL. However, these scenarios assume that participants are honest and respect the protocol. In this last scenario, we demonstrate how FL can be vulnerable to malicious participants, whose goal is to degrade the performance of the global model. To do so, we use poisoning attacks (see Section 2.4.4), where attackers

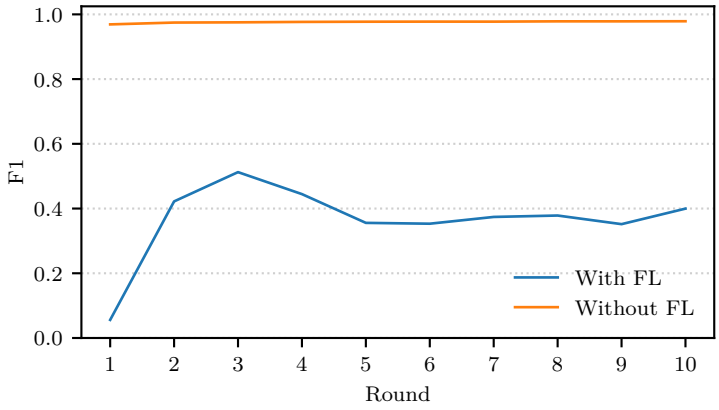


Figure 4.3 – Global model performance in NIID (different sources).

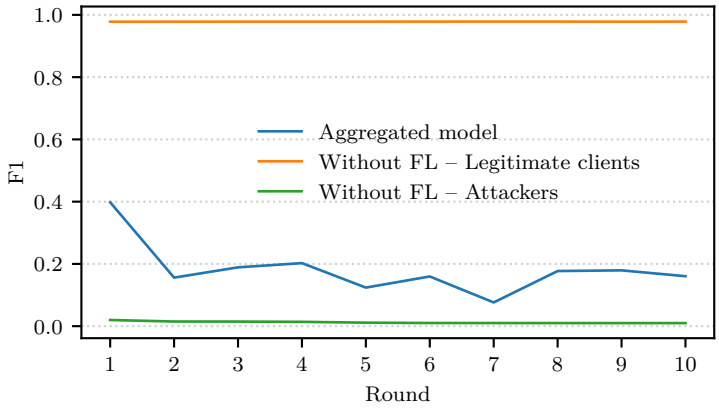


Figure 4.4 – Global model performance in poisoning attacks.

flip the labels of samples in their training data to degrade the performance of the global model.

In order to observe the impact in an extreme scenario, two of the four clients are instructed to perform a label flipping attack on their entire training set. We can observe in local training (Figure 4.4) that participants identified as "Attackers" have a very low DR on their test set, as they literally misclassify all of their testing samples. The two benign participants, on the contrary, reproduce the results of Section 4.3.2, with a high DR on their test set.

In FL however, the global model is impacted by the malicious participants, as illustrated in Figure 4.4. The participants cannot converge towards a stable global model, as the malicious participants' updates are too different from the others. Due to the misclassification introduced by the malicious participants, the global model's performance is degraded, and the F1-score oscillates between 0.1 and 0.2. This is critically low, as it means that the aggregated model either misses a lot of attacks and misclassifies a lot of benign samples. A more in-depth analysis of the impact of poisoning attacks on FL is presented in Chapter 5.

4.4 Conclusion and Takeways

In this chapter, we have presented a practical use case for FL in the context of CIDSs. This use case will be used throughout the rest of the manuscript to illustrate the different contributions and results. Based on this use case, we have also exposed some limitations of FIDSs, notably in terms of data heterogeneity and susceptibility to poisoning attacks. We will explore these limitations further in the next chapters: the impact of data heterogeneity in Chapter 7, and the impact of poisoning attacks in Chapter 5. Finally, we will present some solutions to these limitations in Chapter 6 and **??**.

Part II

Addressing Current Limitations of FIDSs

Chapter 5

Assessing the Impact of Label-Flipping Attacks against FIDSs

5.1 Introduction

Because of its distributed nature, Federated Learning (FL) is highly susceptible to various types of threats, such as poisoning and privacy attacks [Rod+23]. Extensive analyses of poisoning attacks in FL have been conducted [Bha+19; Tol+20] and have shown significant impact on performance. However, in critical applications such as Intrusion Detection Systems (IDSs), the performance of the learning algorithm is of utmost importance, as it directly impacts the security of the monitored system. Consequently, the impact of poisoning attacks on FL for IDSs is a critical concern.

Chapter 4 illustrated this concern easily, showing that a simple label-flipping attack can completely compromise the performance of a Collaborative Intrusion Detection System (CIDS) based on FL. This chapter aims to further investigate the impact of label-flipping attacks on Federated Intrusion Detection System (FIDS), and understand the conditions under which these attacks are most effective. While robust approaches have already been proposed [Vy+21; Yan+23; Zha+22b], few studies focus on understanding and quantifying the impact of poisoning attacks on FL for IDSs. In particular, the effects of label-flipping attacks has been overlooked, as no systematic study has been conducted to understand their impact on FL for IDSs to the best of our knowledge.

This work aims at filling this gap by conducting a systematic and quantitative assessment of the impact of label-flipping attacks on FL for IDSs. While simple in nature, label-flipping attacks are particularly interesting as they are easy to implement, even in a *black-boxed* system, and can have a significant impact on the trained global model. Specifically, this study aims at answering the following research questions, building on **??** RQ3 stated in the Introduction of this thesis.

RQ3-1. *Is the behavior of poisoning attacks predictable?*

RQ3-2. *Do hyperparameters influence the impact of poisoning attacks?*

RQ3-3. *Are IDS backdoors realistic using label-flipping attacks?*

RQ3-4. *Is there a critical threshold where label-flipping attacks begin to impact performance?*

RQ3-5. *Is gradient similarity enough to detect label-flipping attacks?*

The content of this chapter is based on our work published at International Conference on Availability, Reliability and Security (ARES) in August 2024 [LBA24b], which has been extended in perspective of a journal submission. The remainder of this chapter is organized as follows. First, **??** details the methodology used to conduct the experiments, with an emphasis on reproducibility. Then, Section 5.3 presents the results of the experiments, answering the research questions. In Section 5.4, we delve into the related work, especially the existing analyses on the impact of poisoning attacks on FL. Finally, Section 5.5 discusses the implications of the results and concludes the paper.

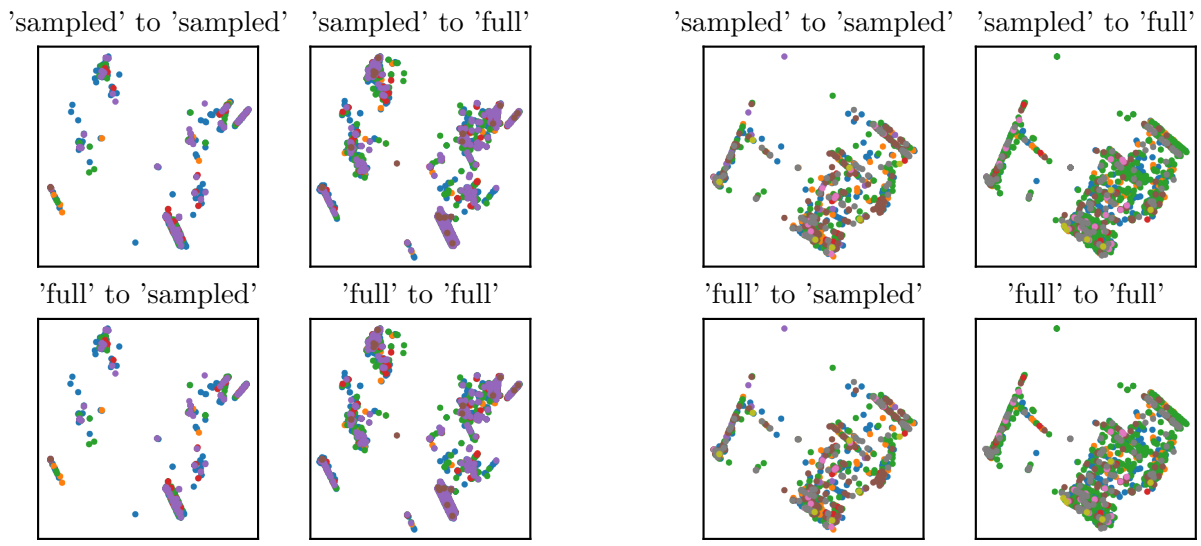
Contributions of this chapter

- The first systematic analysis of the impact of label-flipping attacks on CIDSs leveraging FL, answering a set of well-defined research questions.
- A comprehensive understanding of the impact of these attacks on the performance of the learning algorithm.
- A reusable methodology to assess the impact of poisoning attacks on FL, with experiments that can be easily replicated and extended to other datasets, attack types, and mitigation strategies.
- An automation framework to facilitate the evaluation of FL approaches under different attack scenarios, built upon Flower [Beu+20] and Hydra [Yad19].

5.2 Methodology

Assessing the impact of data-poisoning over FL implies reviewing a consequent amount of parameters and configurations. To optimize our work and make it easily reproducible, the results presented in Section 5.3 have been generated using a purposely designed evaluation framework based on Flower [Beu+20] and Hydra [Yad19]. We follow the ACM's guidelines and terminology [ACM20], and take measures to ensure the *reusability* of our artifacts, the *reproducibility* of our results, and the *replicability* of our experiments. Specifically:

1. We provide the methodology and all parameters necessary to reimplement and replicate the experiments;



(a) CIC-CSE-IDS2018 **(b)** UNSW-NB15

Figure 5.1 – Cross-projections of the malicious traffic from the two datasets in 2D using Principal Component Analysis (PCA). On top, the frame of reference is computed using the `sampled` dataset, and on the bottom the `full` dataset. The `sampled` dataset is then projected on the left, the `full` dataset on the right.

2. Dependencies are pinned using Poetry for Python and Nix for system, allowing the entire software pipeline to be executed in the same conditions;
3. All experiments are seeded where possible, which makes the results reproducible within a three decimal precision;
4. The results and the code to generate them are available in open access [1], as are the datasets [2].

The results presented in this paper amount to 10 670 unique runs, and close to 1 613 cumulated computing hours on two NixOS servers with 96 cores, 768 GB of RAM and 2 Nvidia Tesla T4 each.

5.2.1 Dataset and Pre-processing

Due to the scale of the required experiments, we require a datasets that are both representative of the problem and small enough to be processed in a reasonable amount of time. As discussed in Chapters 2 and 4, recent works on FL and IDS [SLP21] proposed a standardized feature set (NF-V2) making cross-dataset FL setups easier. The authors notably provide converted versions of known datasets, including CSE-CIC-IDS2018 [SHG18] and UNSW-NB15 [MS15], the two most used generic Network-based Intrusion Detection System (NIDS) datasets in the literature.

1. `httos://github.com/TODO`
2. `https://staff.itee.uq.edu.au/marius/NIDS_datasets/`

Table 5.1 – Distribution of the CIC-CSE-IDS2018 and UNSW-NB15 (NF-V2) datasets [LP22; SLP22].

Class	Sampled	Full	Class	Sampled	Full
Benign	880,623	16,635,567	Benign	960,078	2,295,222
DDoS	73,558	1,390,270	Exploits	13,187	31,551
DoS	25,574	483,999	Fuzzers	9,377	22,310
Bot	7,595	143,097	Generic	6,976	16,560
Brute Force	6,525	123,982	Reconnaissance	5,352	12,779
Infiltration	6,108	116,361	DoS	2,455	5,794
Injection	17	432	Analysis	969	2,299
			Backdoor	925	2,169
			Shellcode	617	1,427
			Worms	64	164
Total	1,000,000	18,893,708	**Total**	1,000,000	2,390,275

Due to the scale of the required experiments, we use the "sampled" version of the datasets provided by the authors [LP22], and already mentioned in Chapter 4. After pre-processing, we evenly split the dataset for the experiments, ensuring the same class distribution in the training and testing sets. 80% of the dataset is used for training, and 20% for testing. We purposely do not use a proper validation set, as the goal is to measure the performance delta related to the impact of the attacks on the global model, and not to optimize the model's hyperparameters. Indeed, the parameters are kept constant throughout the experiments, as detailed in Table 5.2. We sometimes refer to the datasets as `cicids` and `nb15` for brevity in the remaining of this chapter.

To assess the representativity of the datasets sampling, we compare the projections in two dimensions of the two datasets using PCA. Figure 5.1 presents cross-projections results, depending on the datasets used to generate the projection frame. There are consequent overlaps between the classes in this projection, implying that either 2 dimensions are not enough to separate the classes, or there are features that are not relevant to the classification task. Yet, the projected patterns are identical between the two datasets, which indicates that the sampling process does not introduce significant distribution bias in the dataset. Therefore, experiments performed over the sampled datasets should be representative of observed the behaviors in the original dataset.

5.2.2 Local and Federated Training

We use a simple Multilayer Perceptron (MLP) model with two hidden layers, as implemented by Popoola, Gui, *et al.* [Pop+21b] who use the same datasets; a summary of the model's parameters is available in Table 5.2. Trained centrally, this model reaches an F1-score of 0.966 and an accuracy of 0.992 on our sampled CIC-CSE-IDS2018, and 0.945 and 0.995 on the UNSW-NB15 dataset, respectively. These values can be considered as

Table 5.2 – Hyperparameters.

Hyperparameter	Value
Learning rate	0.0001
Hidden layers activation	ReLU
Output layer activation	Sigmoid
Input shape	49
Number of hidden layers	2
Size of the hidden layers	128
Optimizer	Adam
Loss function	Binary cross-entropy
Aggregation	`FedAvg`

baselines for the FL experiments.

We focus on the impact of data-poisoning specifically, and therefore omit other factors that could hurt the performance of the model, such as client heterogeneity or disconnections. We also specifically concentrate our efforts on a collaborative cross-silo setting, where all clients are available at each round and $C = 1$. Consequently, the dataset is partitioned into 10 Independent and Identically Distributed (IID) shards of 80,000 data points, and each client is assigned with one shard. On the server, the uploaded models are aggregated using `FedAvg`—which, since the local datasets are of similar size, is equivalent to a simple average of the weights.

5.2.3 Attack Model and Implementation

We consider data-poisoning attacks where malicious participants can alter their local datasets before training. This definition covers both, participants that have been compromised and those that are deliberately modifying their data. Further, this scenario will always be available, even with a secure and immutable FL client software. Specifically, we implement data-poisoning using label-flipping attacks, where the attacker changes the label y of a sample to a new label y_p; *i.e.*, $y_p = \neg y$ in a binary-classification problem.

Attacker's Objective We consider two types of objectives for the attacker depending on the type of attack leveraged. With *targeted* attacks, the attacker aims to make a specific attack pattern undetectable, and therefore act as a backdoor in the IDS. This is implemented by labeling a randomly selected fraction of a specific attack class (*e.g.*, *DDoS*) as benign. With *untargeted* attacks, on the other hand, his goal is to produce high False Positives Rate (FPR) and False Negative Rate (FNR), which can overwhelm human operators or other security systems. Here, a random fraction of the entire dataset is altered, where the label of each sample is flipped from benign to attack and vice versa. The proportion of samples that are altered is controlled by the Data Poisoning Rate (DPR),

Table 5.3 – Experimental parameters. Default parameters are highlighted in bold and are used if not specified otherwise.

Parameter	Values
dataset	cicids, nb15
batch_size	32, 128, **512**
epochs	**100_10x10**, 100_4x25, 100_1x100, 300_10x30, 300_4x75, 300_1x300
distribution	10-0, 9-1, 7-3, **5-5**, 3-7
scenario	continuous-{10,30,60,70,80,90,95,99}, **continuous-100**, late-3, redemption-3
target	**untargeted**, (cicids) bot, dos, ddos, bruteforce, infiltration, injection, (nb15) generic, analysis, worms, backdoor, exploits, untargeted, shellcode, dos, reconnaissance, fuzzers
partitioner	iid_drop_1, iid_drop_2, **iid_full**, iid_keep_1, kmeans_drop_1, kmeans_drop_2, kmeans_full, kmeans_keep_1
seed	1313, 1977, 327, 5555, 501, 421, 3263827, 2187, 1138, 6567

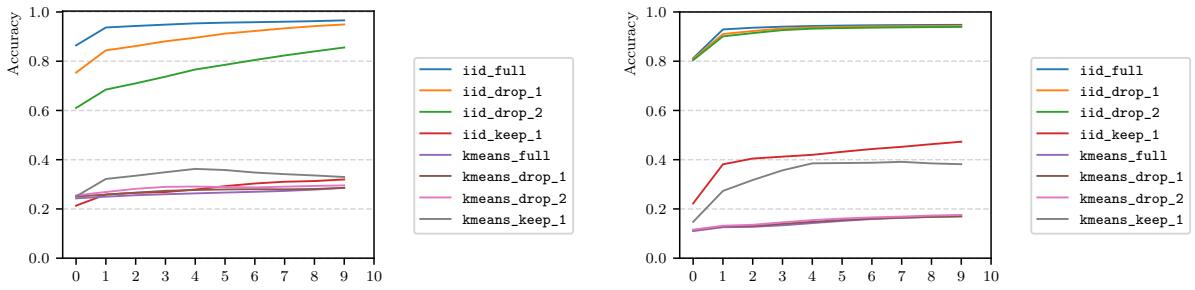
which is the ratio of samples matching the target that are altered by each attacker on a specific round. We note the DPR, or *local poisoning rate*, as α.

Attacker's Knowledge and Capabilities We consider attackers to be *gray-box* adversaries, *i.e.*, they have the same knowledge as benign clients, but are unable to modify the system's behavior, neither locally nor on the server. Further, we consider that multiple attackers can be present in the system, and that they can act in concert. This scenario is referred to as *colluding attackers*. In this case, the attackers share the same target and DPR. The proportion of attackers can vary from one single malicious client to a majority of them being malicious, and is expressed as τ, or Model Poisoning Rate (MPR) [Mer+23]. Note that in the context of IID partitioning, the overall poisoning rate could be regarded as $\alpha \times \tau$. This simplification is however not accurate in other partitioning strategies.

5.2.4 Experiments

We design a set of experiments to answer the research questions laid out in Section 5.1. All experiments share a common set of constants, which are complemented by a set of variable parameters. Table 5.3 summaries the available parameters for the experiments. Each combination is tested 10 times using a set of 10 different seeds to study the predictability of the results. Specifically, the seed impacts data-partitioning operations (both between the training and testing sets, and among clients afterward), the sample selection in poisoning attacks, and the random weights of the initial model. It also impacts all the random operations (such as data shuffling) done during model training.

The `epochs` parameter controls the aggregation frequency, *i.e.*, the number of local epochs per round $\mathcal{E}$, as well as the number of rounds R. Logically, `batch_size` controls



(a) CIC-CSE-IDS2018 **(b)** UNSW-NB15

Figure 5.2 – Impact of the partitioning strategy on the model's performance.

the size of the batches used at each iteration during training. The global number of local epochs per client is kept to 100 or 300 to preserve comparability. The `distribution` represents the number of legitimate and malicious clients in the system, and consequently the proportion of attackers. The key `scenario` represents the attackers' behavior. Scenarios defined as `continuous-`α represent a constant poisoning rate of α over the entire training process. Scenarios named `late-`r and `redemption-`r produce an attack with $\alpha = 100$ that starts or ends at round r, respectively. Parameter `target` represents the target of the attack as defined in Section 5.2.3; each attack class is made available as a target.

5.2.4.1 Partitioning Strategies

A particular parameter in this selection is the `partitioner`, which defines the strategy used to partition the dataset among the clients. As mentioned in Section 4.3.1, the literature of FIDSs usually considers either IID partitioning or basic Non Independent and Identically Distributed (NIID) scenarios where clients randomly drop attack classes. To study the usage of similarity metrics to detect poisoning attacks, we consider eight different partitioning strategies, with different levels of heterogeneity that range from IID partitioning to *pathological*[3] settings without distribution overlap.

The `partitioners` marked named `iid_*` distribute the data in an IID manner, where each client receives a stratified [4] subset of the dataset. Then, `full`, `drop_*`, and `keep_*` refer to the way the attack classes are processed: all kept, certain classes dropped, or certain classes kept, respectively. The `kmeans_*` partitioners use a k-means clustering algorithm to partition the benign traffic among the clients, and then distribute the attack classes based on their attribute (`full`, `drop_*`, `keep_*`). Because k-means algorithm minimizes the intra-cluster variance, it increases the heterogeneity of the clients' datasets. Figure 5.2 illustrates performance of the model under different partitioning strategies.

3. Refers to the *pathological* NIID partitioning [McM+17] discussed in Chapter 2.
4. Stratified sampling ensures that the class distribution remains the same among the different shards. It is usually done using the labels of the samples, but any categorical feature can be used.

5.2.5 Metrics

To quantify how the experiment parameters impact the global model, we define a set of metrics to measure the Attack Success Rate (ASR) of poisoning attacks. The definition of the ASR differs depending on the type of attack, according to the attacker's objective defined in Section 5.2.3. Because the ASR is based on performance and that no perfect model exists, we distinguish the Absolute Attack Success Rate (AASR) measured on the attack scenario, from the Relative Attack Success Rate (RASR) which also considers the nominal performance without attacks. Formally, the RASR is defined as:

$$\text{RASR} = \frac{\max(\text{AASR}_{benign}, \text{AASR}_{attack}) - \text{AASR}_{benign}}{1 - \text{AASR}_{benign}}, \tag{5.1}$$

where AASR_{benign} and AASR_{attack} are the AASR of the *benign* and *attack* scenarios respectively, under the same set of parameters. This is made possible thanks to the framework's reproducibility, which ensures two experiments started with the same seed will run under the same conditions. Following the definitions in Section 5.2.3, we then defined two variations of the AASR depending on the attacker's objective. Both are computed based on the confusion matrix of the model: True Positive (TP), True Negative (TN), False Positive (FP), and TN.

Targeted attacks: Malicious participants leverage targeted attacks to make a specific attack pattern undetectable. Therefore, a successful attack forces classification of the relevant attack samples as benign. The AASR is then defined as the miss rate of the targeted attack, *i.e.*

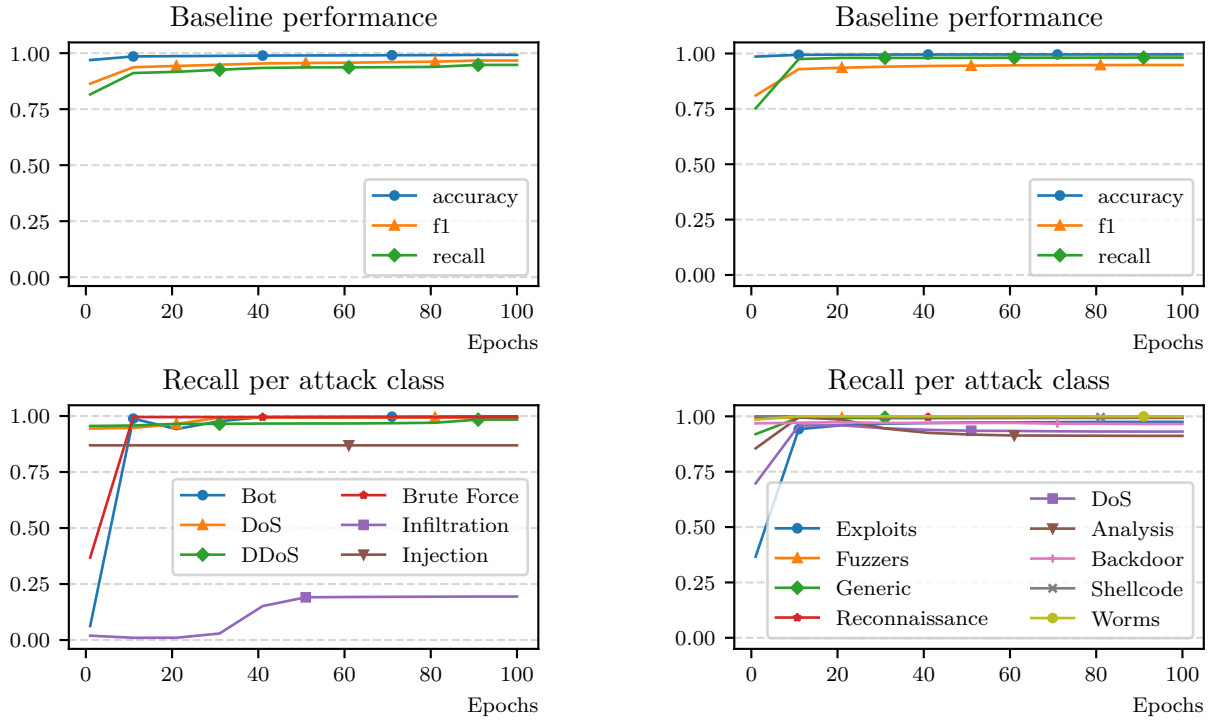
$$\text{AASR} = \frac{\text{FN}_c}{\text{TP}_c + \text{FN}_c}, \tag{5.2}$$

where c is a specific attack class of the dataset.

Untargeted Attacks: Untargeted attacks aim at degrading the overall classification rate of the model. Consequently, the AASR is defined as the miss-classification rate of the model, *i.e.*

$$\text{AASR} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = 1 - \text{accuracy}. \tag{5.3}$$

Additionally, we use traditional binary classification metrics to observe the performance of the model under various conditions, as identified in existing surveys [Cam+22; Lav+22b] These metrics include *accuracy*, *F1-score*, and *miss rate*. Notably, we consider the Main Task Accuracy (MTA), defined as the accuracy of the benign clients obtained on the testing set, to measure the impact of the attacks on the model's nominal performance. All metrics are aggregated over the 10 runs of each experiment, and the mean and standard deviation are reported for the selected metric.



(a) CIC-CSE-IDS2018. **(b)** UNSW-NB15.

Figure 5.3 – Baseline performance of the global model without malicious participants. The accuracy, F1-score, and recall illustrate the performance that can be expected from the global model under the conditions selected for this study ($\mathcal{E} = 10$, $\beta = 512$). The recall of the six available attack classes shall serve as a reference for the RASR of targeted attacks.

5.3 Results

The results presented in this section aim at answering the research questions defined in Section 5.1. Figure 5.3 presents the performance of the global model without malicious participants to serve as a baseline to compare with. The model displays relatively high performance on both datasets, *i.e.*, above 0.9 for the accuracy, recall, and F1-score. However, some classes are more challenging to detect than others. The feeble representation of the "Injection" class in `cicids` (around 0.0017%, see Table 5.1) prevents the model from learning from it, provoking this absence of evolution over time (see Figure 5.3a). The "Infiltration" class is more represented in the dataset (0.6108%, approximately the same as the "Brute Force" and "Bot" classes), but remains difficult to learn because of its apparent similarity with benign traffic. Consequently, it never exceeds a recall of 0.2. The detection in `nb15` is better overall, with the exception of the "Analysis" and "DoS" classes that score below 0.9, although they are not the least represented classes in the dataset.

Table 5.4 – Experiment parameters for RQ3-1..

Is the behavior of poisoning attacks predictable?	
batch_size	32, 512
epochs	300_10x30, 300_4x75, 300_1x300
distribution	5-5
dataset	cicids, nb15

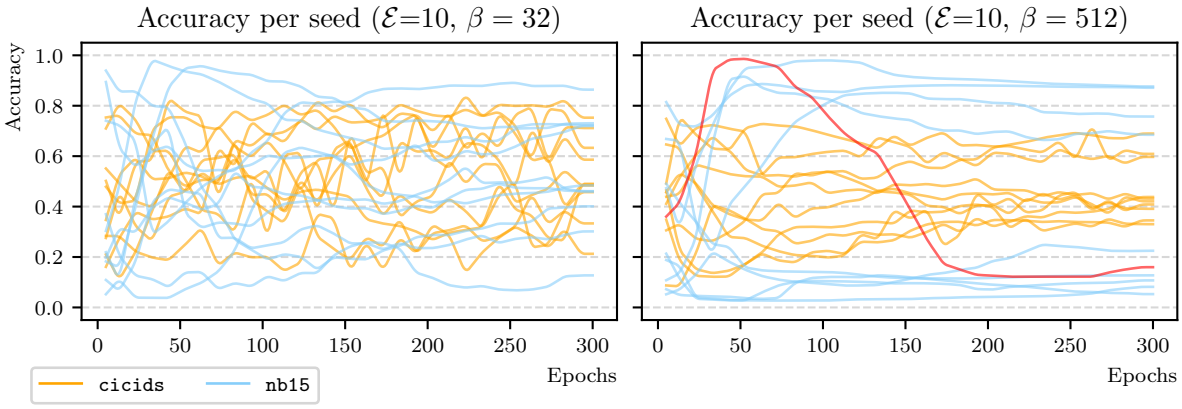


Figure 5.4 – Impact of the dataset on the accuracy under poisoning. $\tau = 0.5, \mathcal{E} = 10$. The x-axis represents the number of local epochs. Each line represents the accuracy over time for a specific seed. The seed 6567 on `nb15` ($\beta = 512$) is depicted in red.

5.3.1 Impact Predictability

A preliminary question to answer before quantifying the effects of label-flipping is whether the behavior of poisoning attacks is predictable. This is a requirement for generalizing our results to other datasets and models, and comparing the findings with current and future studies. Due to space constraints and computation constraints, we focus in this part on the parameters that have the most significant impact on the results, as we need to perform longer experiments to ensure the stability of the results. Specifically, the selected distribution contains 50% of malicious participants (*i.e.*, $\tau = 0.5$), which roughly equates to 50% of the training data being poisoned. The experiments are performed during 300 epochs, with three different aggregation frequencies ($\mathcal{E} \in \{1, 4, 10\}$) and two different batch sizes ($\beta \in \{32, 512\}$). Table 5.4 summarizes the parameters used for this experiment.

Figure 5.4 illustrates the accuracy of the global model over time for each seed. This figure brings two key insights. First, the different runs exhibit consequent variability, especially in the early stages of training. This is true for the runs themselves, but also for between runs on the same datasets. Using the same parameters, the accuracy of the global model varies from 0.2 to 0.8 after 100 epochs on `cicids` (with $\beta = 32$) and from 0.05 to 0.95 on `nb15`. Some runs display particularly heterogeneous results, such as the seed 6567

Table 5.5 – Experiment parameters for RQ3-2..

Do hyperparameters influence the impact of poisoning attacks?	
batch_size	32, 128, 512
epochs	100_10x10, 100_4x25, 100_1x100
distribution	10-0, 5-5
scenario	continuous-100, late-3, redemption-3

on nb15 ($\beta = 512$), which starts below 0.4, increases to close to 1.0, before dropping below 0.2 and stabilizing. Overall, nb15 presents a higher variability than cicids, but with runs that are more stable.

The dispersion decreases over time given a big enough batch size, as shown in Figure 5.5. More importantly, each runs progressively converges to a stedier state. The absolute accuracy differences in the bottom rows of Figure 5.5 indeed decrease over the first epochs, before plateauing. It can be interpreted as a consequence of the complexity of the learning tasks, which becomes harder as clients possess different labels for similar samples. Therefore, the problem probably admits a high number of local minima, which are reached depending on the seed. On the contrary, the difference between rounds using $\mathcal{E} = 32$ on cicids (see Figure 5.5a) tends to increase over time, illustrating the difficulty for each run to converge to a stable state.

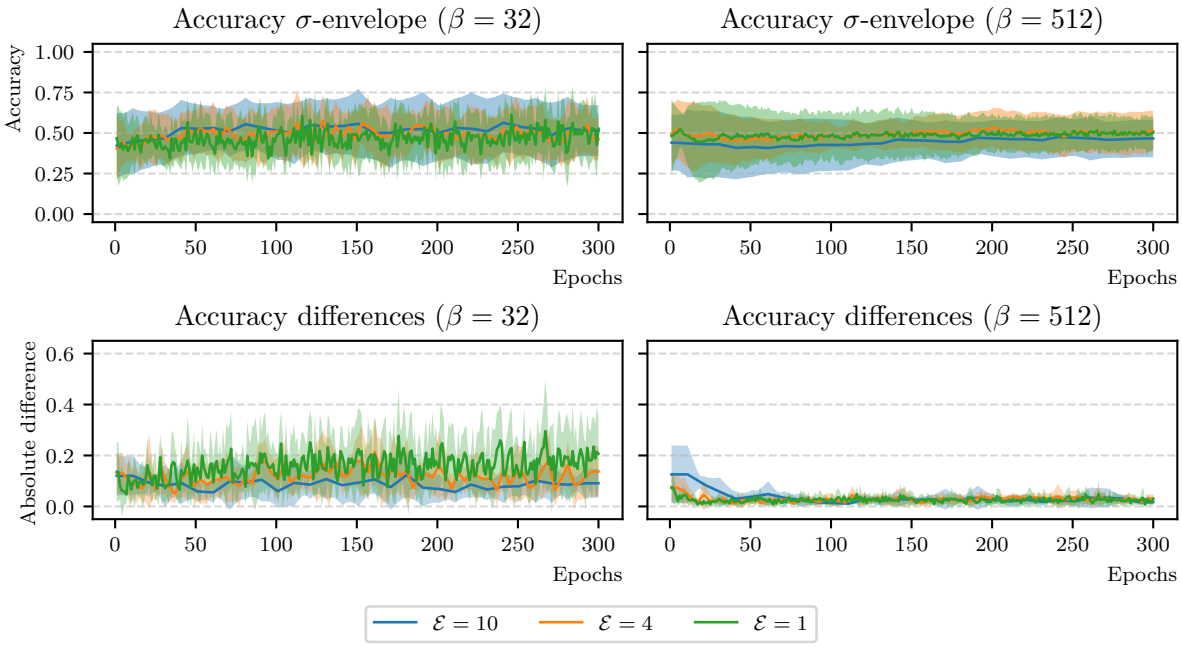
Answering RQ3-1.

The behavior of poisoning attacks is not predictable, as the dispersion between results is too important, although only the seed varies. However, the dispersion decreases over time in some conditions, and each run tends to converge to a stable state. *In practice, this makes the impact difficult to predict for a specific attack instance, even though general tendencies can be extrapolated.*

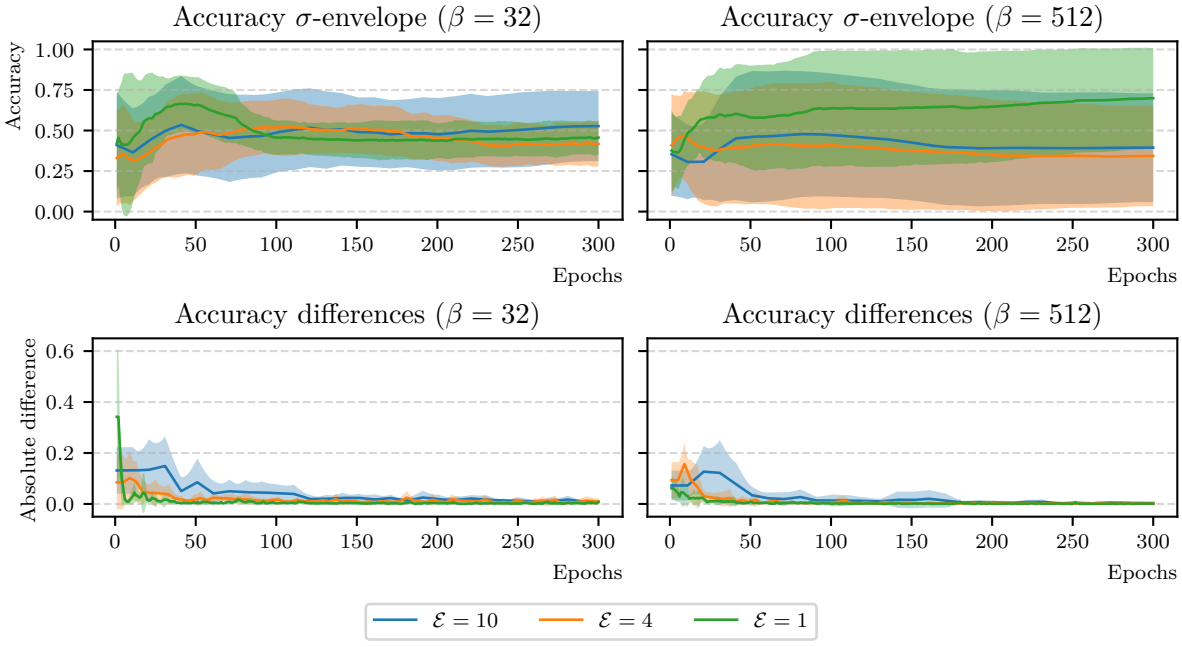
5.3.2 Hyperparameters Impact

To understand the impact of hyperparameters on the behavior of poisoning attacks, we study the impact of different batch sizes (β) and aggregation frequencies ($\mathcal{E}$). We reuse the conditions from Section 5.3.1 but limit the number of epochs to 100, as most scenarios do not show significant changes after this point (see Section 5.3.1). Additionally, we evaluate the hyperparameters on the *late* poisoning scenario, where the attackers only start after a 3-rounds bootstrap period, and in the *redemption* scenario, where the attackers stop after 3 rounds. The experiment parameters are summarized in Table 5.5.

Figure 5.6 illustrates the influence of hyperparameters on the impact of poisoning attacks. The differences (the two bottom rows) are shown on a bi-symmetric logarithmic

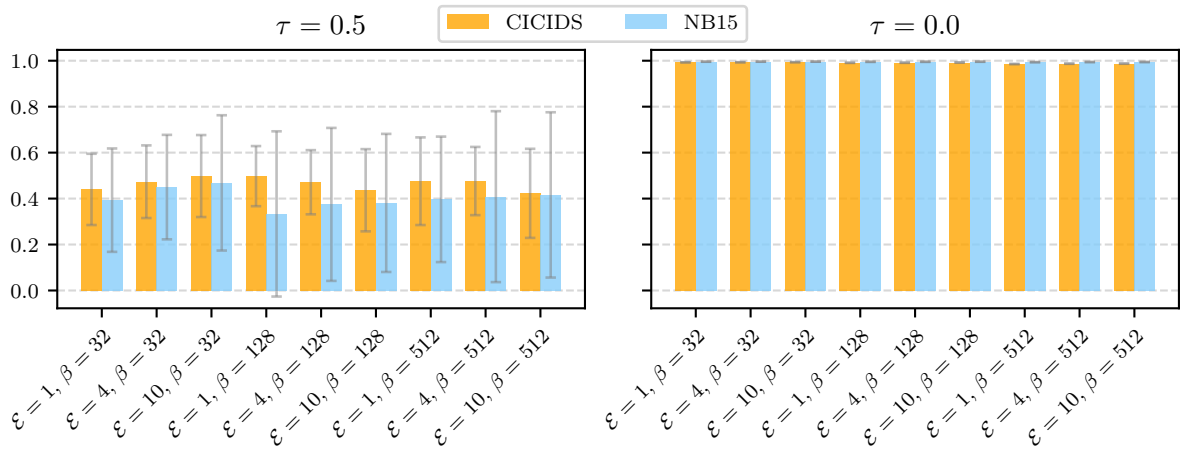


(a) CIC-CSE-IDS2018.

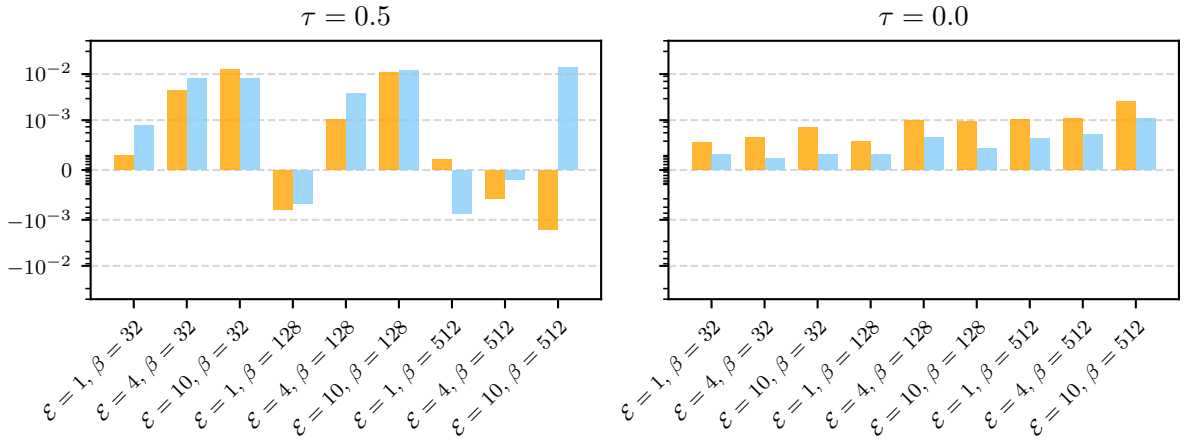


(b) UNSW-NB15.

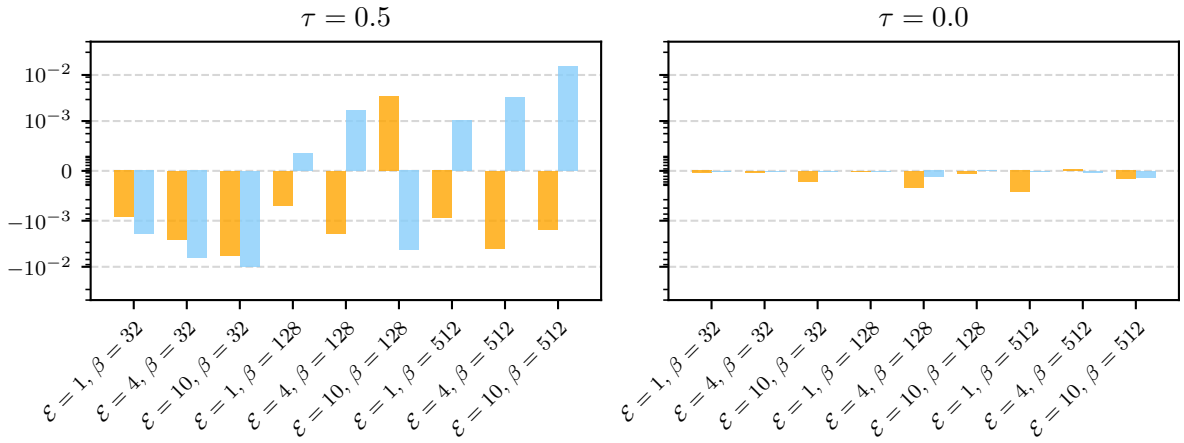
Figure 5.5 – Studying attack impact predictability over time, with 50% attackers. The x-axis represents the number of local epochs. For each subfigure (Figures 5.5a and 5.5b), the top row illustrates each seed's accuracy over time with its standard deviation. The bottom row displays the mean absolute difference ($|\text{acc}_r - \text{acc}_{r-1}|$) over time.



(a) Mean accuracy



(b) Mean accuracy difference ($x_r - x_{r-1}$)



(c) Mean σ difference ($\sigma_r - \sigma_{r-1}$)

Figure 5.6 – Comparing the influence of hyperparameters on the impact of label-flipping attacks ($\tau \in \{0, 0.5\}$ and $\alpha = 1$). The x-axis represents the number of local epochs. The top row illustrates the mean accuracy for each combination of hyperparameters (β and $\mathcal{E}$). The middle row displays the mean difference in standard deviation (σ) between rounds. The bottom row shows the mean difference in terms of accuracy between rounds.

scale [Web12], defined as

$$x' \mapsto \text{sgn}(x) \cdot \log_{10}(1 + |\frac{x}{10^{-4}}|), \tag{5.4}$$

to make up for the consequent differences in scale between combinations. In addition to the accuracy of each parameter combination, Figure 5.6 also presents the average change in standard deviation, or

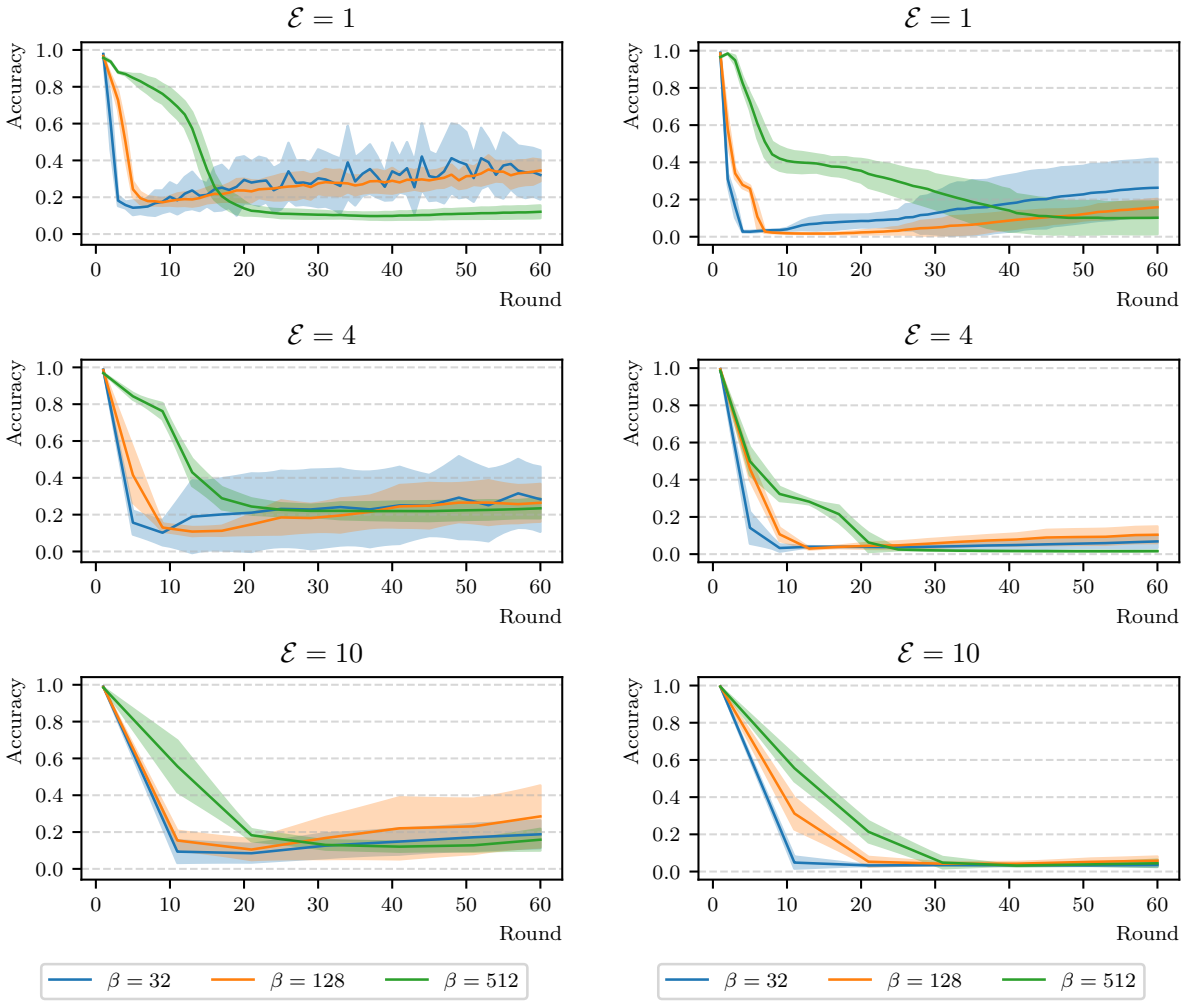
$$\frac{1}{R-1} \sum_{r=2}^{R} \sigma^r - \sigma^{r-1}, \tag{5.5}$$

where R is the number of rounds and σ^r is the standard deviation of the accuracy at round r between the different seeds. The average change accuracy is also displayed. For these metrics, a positive value indicates an increase in the observed metric over time.

In the continuous scenario studied in Figure 5.6 (with $\tau = 0.5$ and $\alpha = 1$), the hyperparameters have little impact on the global model's accuracy. Yet, $\beta = 32$ presents a slight increase in accuracy over time (Figure 5.6b) and a decrease in the dispersion (Figure 5.6c). Note that there is close to no dispersion in the accuracy of the benign scenario, as depicted in Figure 5.6c, confirming that the attack is indeed responsible for the dispersion observed in Section 5.3.1. Interestingly, the correlations between the two tested datasets are more pronounced with smaller batch sizes, as shown in Figures 5.6b and 5.6c where the results become less correlated with $\beta = 512$.

The other hyperparameters do not display any significant correlation. In the end, all tested parameters lead to between 0.4 and 0.5 accuracy under poisoning, while they all exceed 0.95 without poisoning. This is critically low for intrusion detection: 0.5 is the score of a random classifier on a balanced binary-classification task. *Tossing a coin would yield better results.*

However, when clients have been given the time to converge before the attack, the impact of the hyperparameters becomes more visible, particularly for the batch size as depicted in Figure 5.7. While the impact is instantaneous when $\beta = 32$, it takes around 20 epochs with $\beta = 512$ to reach the same accuracy. The dispersion of the results is significantly lower in the latter, as is the reached accuracy, which goes down to 0.25 after 60 epochs. A bigger batch size thus leads to a greater inertia and a lower dispersion of the results when the attack starts, but also to a lower accuracy afterward. A similar could be observed with the `redemption` scenario, where the attack stops after 3 rounds. However, he results show a quasi-instantaneous recovery of the global model's accuracy after the attack ends. This is expected, as Figure 5.3 indicates that the global model's accuracy already exceeds 0.95 at the first round, in spite of the randomly initialized model parameters provided by the server before the first round. This is also consistent with the results of Zhang, Zhang, Zhang, *et al.* [Zha+22a] on NSL-KDD [Tav+09] and UNSW-NB15 [MS15].

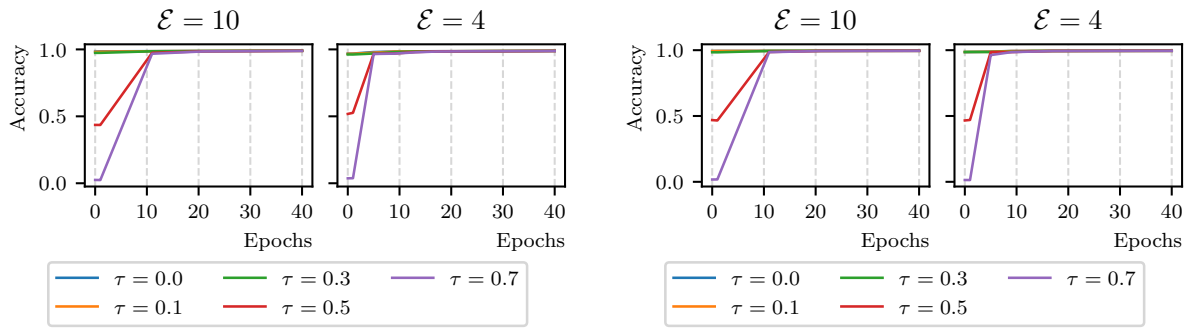


(a) CIC-CSE-IDS2018. **(b)** UNSW-NB15.

Figure 5.7 – Impact of hyperparameters on the accuracy of the global model under the `late` scenario. The data is aligned to start at the last benign round before the attack, and the impact is measured over the next 60 epochs (*i.e.*, 6, 24 or 60 rounds depending on the aggregation frequency).

Answering RQ3-2.

While the hyperparameters have an impact on the poisoning effect, no combination prevents it: *on average, the performance remains the same.* The results' dispersion can vary significantly depending on parameter combinations, especially when the attack occurs after the clients have converged. Then, a smaller batch size leads to a swifter effect, while a bigger batch size leads to a greater ASR. *Therefore, in performance-constrained use cases (such as the Internet of Things (IoT)), defense mechanisms might need to react faster to mitigate the attack's impact.* Round-based defenses should be less affected, but history-based defenses could be significantly impacted. When the attack stops, the global model's accuracy recovers almost instantaneously.



(a) CIC-CSE-IDS2018. **(b)** UNSW-NB15.

Figure 5.8 – Accuracy of the global model after a label-flipping attack. The data is aligned to start at the last epochs before the attack ends, and the impact is measured over the next 40 epochs (*i.e.*, 4 or 10 rounds depending on $\mathcal{E}$).

Table 5.6 – Experiment parameters for RQ3-3..

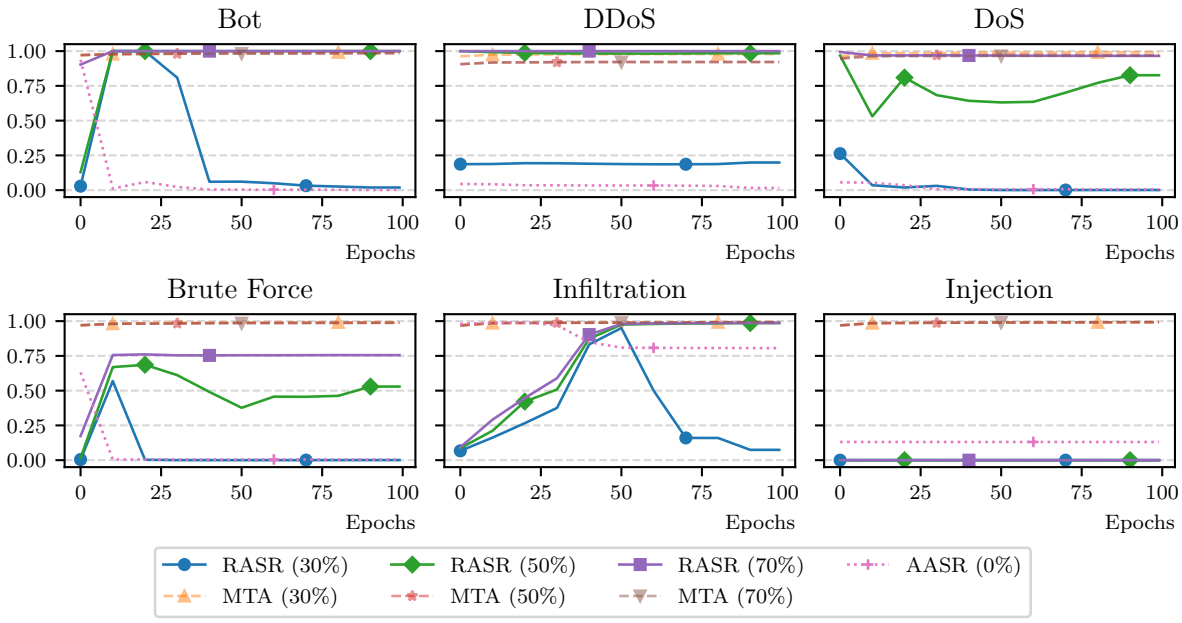
Are IDS backdoors realistic using label-flipping attacks?	
`distribution`	`10-0, 7-3, 5-5, 3-7`
`target`	`dos, ddos, bot, infiltration, injection`
`scenario`	`continuous-100`
`dataset`	`cicids, nb15`

5.3.3 IDS Backdoors using Label-flipping

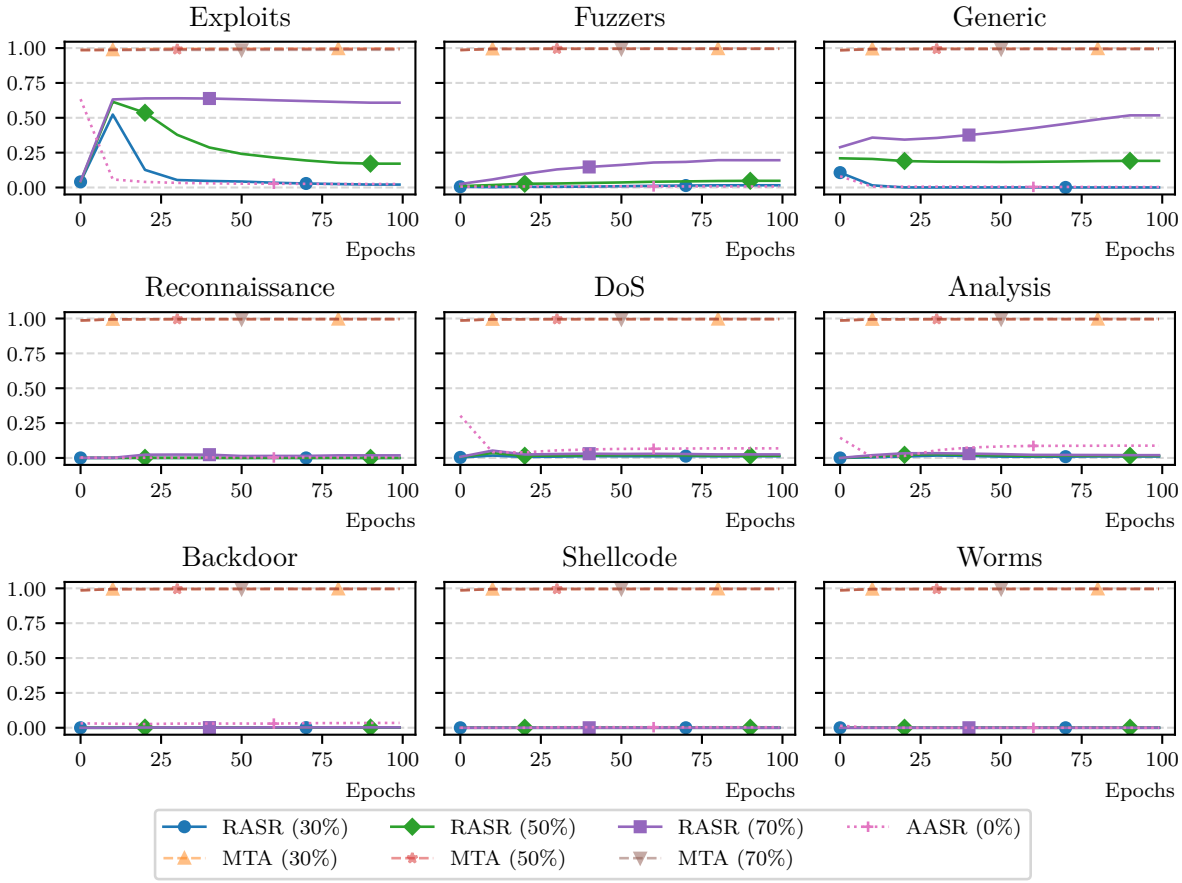
One of the main concerns with poisoning attacks is the perspective of backdoors in the IDS, allowing attackers to bypass the system's detection capabilities afterward. To assess this risk, we study the impact of label-flipping attacks with different targets. We consider $\alpha = 100\%$ and various values of τ to assess whether a ASR of 1.0 can be achieved. Table 5.6 summarizes the parameters used for this experiment.

Figure 5.9 presents the impact of label-flipping attacks on the accuracy of the global model for different attack targets. The figures show the RASR of targeted attacks towards the different classes available in each dataset. To measure the impact of the attack on the model's overall performance, we also measure the MTA under the differnet attacks. The AASR of the benign scenario is provided as a reference (*c.f.* Figure 5.3).

The first striking observation is that the results differ greatly between the two datasets. While on `cicids`, some attacks can reach an ASR close to 1.0 given enough attackers, most results on `nb15` remain below 0.25. Some classes seem completely immune to the attack, such as "Backdoor", "Shellcode", and "Worms". Even the most successful attacks on `nb15` barely exceed 0.5. On `cicids`, on the other hand, the results are more convincing. hile 30% of attackers are not enough to permanently impact the global model's accuracy, multiple classes can reach an RASR close to 1.0 at least momentarily. With $\tau = 0.5$, the RASR of "Bot", "DDoS", and "Infiltation" classes permanently reach 1.0 after a few



(a) CIC-CSE-IDS2018.



(b) UNSW-NB15.

Figure 5.9 – RASR of targeted label-flipping attacks over time, with $\beta = 512$, $\mathcal{E} = 10$, and $\alpha = 100\%$. The x-axis represents the number of local epochs. The AASR of the benign scenario is provided as a reference for each targeted class.

Table 5.7 – Experiment parameters for RQ3-4..

Is there a critical threshold where label-flipping attacks begin to impact performance?	
`distribution`	`10-0, 9-1, 7-3, 5-5, 3-7`
`scenario`	`continuous-{10,30,60,70,80,90,95,99,100}`
`target`	`untargeted, dos, ddos, bot, infiltration`
`dataset`	`cicids, nb15`

rounds. Most importantly, the MTA of the global model remains close to its nominal performance, even when the RASR of the targeted classes reaches 1.0, except for the "DDoS" class. Indeed, the "DDoS" class is the most represented in the dataset, with 5.29% of the samples. Therefore, the misclassification of roughly 70% of the samples of this class leads to a more significant impact on the global model's accuracy.

The result disparity between classes, and more importantly between datasets, is difficult to interpret, although multiple hypotheses can be formulated. First, some classes are significantly less represented in the datasets such as "Injection" in `cicids` or "Analysis", "Backdoor", "Shellcode", and "Worms" in `nb15` (see Table 5.1). All these targets yielded a RASR close to 0.0. Second, as we consider binary classification tasks, any overlap between classes' characteristics can be used by the model to infer the correct associations using samples from unaffected classes. For instance, the "Brute Force" and "DoS" classes in `cicids` both imply a high number of connections from a single host, and both obtain subpar results when compared to the most efficient attacks.

Answering RQ3-3.

This type of attack has less impact on the global model's MTA, meaning that they are more likely to remain undetected. Although not all classes are equally impacted, *IDS backdoors are possible using label-flipping attacks, given a sufficient number of attackers and a well-represented target.* Colluding attackers can realistically create a backdoor that may later be leveraged to evade detection, raising the question of the minimum DPR and MPR necessary for such attacks to be effective. Yet, the results are dataset-dependent, and some classes remain completely immune to the attack in our experiments.

5.3.4 Threshold for Effective Attacks

Our previous experiments highlighted the impact of the number of attackers on the global model's accuracy. Section 5.3.3 suggests that the number of attackers is a critical factor in the effectiveness of targeted attacks. This experiment aims to understand the

critical threshold where label-flipping attacks begin to impact the global model's accuracy by studying both the DPR (α) and the MPR (τ). Table 5.7 summarizes the parameters used for this experiment.

?? presents the RASR of considered label-flipping attacks over time, both for untargeted and targeted attacks. The entire figure emphasizes on the importance of the number of attackers in the effectiveness of the attack, as most attacks are unimpactful with while $\tau \leq 0.3$. In particular, with one single attacker ($\tau = 0.1$), the RASR remains well below 0.1, even with a high poisoning rate ($\alpha = 1.0$). $\tau = 0.5$ represents a tipping point in our tests, with RASR values orders of magnitude higher than with $\tau = 0.3$.

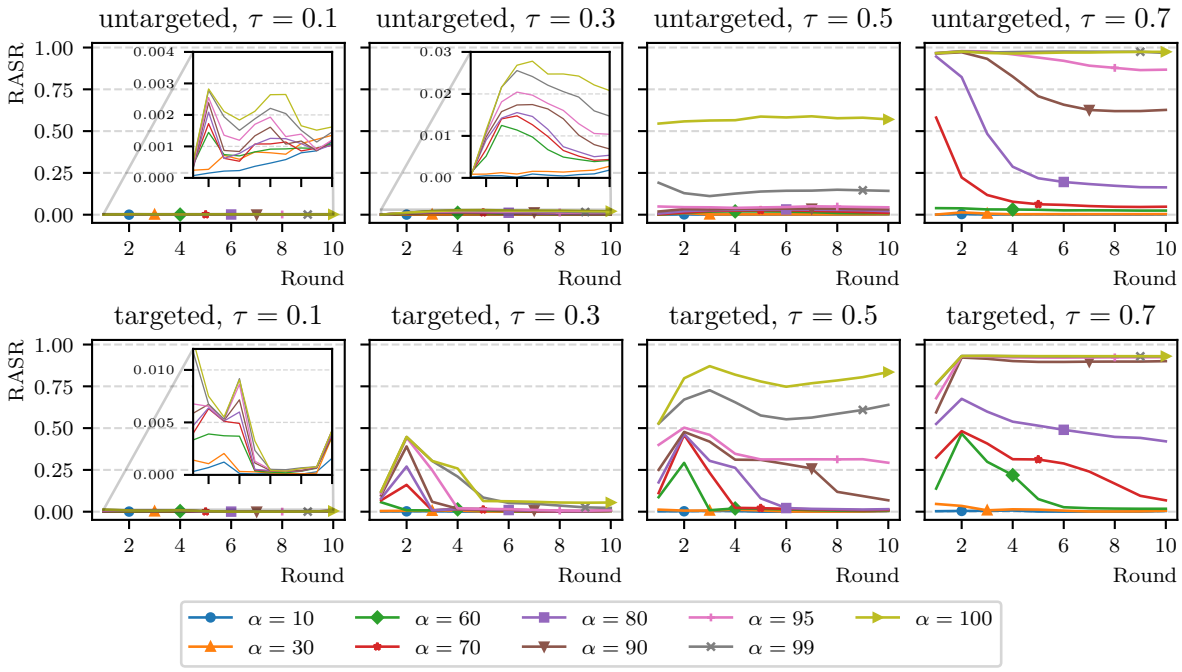
More importantly, we can infer from $\Gamma = \tau \times \alpha$ the overall quantity of poisoned data due to our IID partitioning. For untargeted attacks in `cicids`, the RASR exceeds 0.5 for $\Gamma > 0.5$, and approach 1.0 for $\Gamma > 0.67$. For targeted attacks, the RASR exceeds 0.5 for $\Gamma > 0.49$ and approaches 1.0 for $\Gamma > 56\%$. Untargeted attacks on `nb15` are slightly more effective, with the RASR exceeding 0.5 for $\Gamma > 0.49$ and approaching 1.0 for $\Gamma > 0.63$. However, as highlighted in Section 5.3.3, targeted attacks on `nb15` are close to ineffective, with the RASR never exceeding 0.25, even with $\Gamma = 0.7$, our highest overall poisoning rate. Thus, RASR and the tuple (α, τ) exhibit fairly similar variations, albeit not linear: the higher are the DPR and MPR, the higher is the RASR. However, poisoning the entire local dataset seems more powerful than instantiating more attackers: $\alpha = 100$ and $\tau = 50$ yield higher RASR than $\alpha = 80$ and $\tau = 70$, although the latter represents more affected data overall.

Answering RQ3-4.

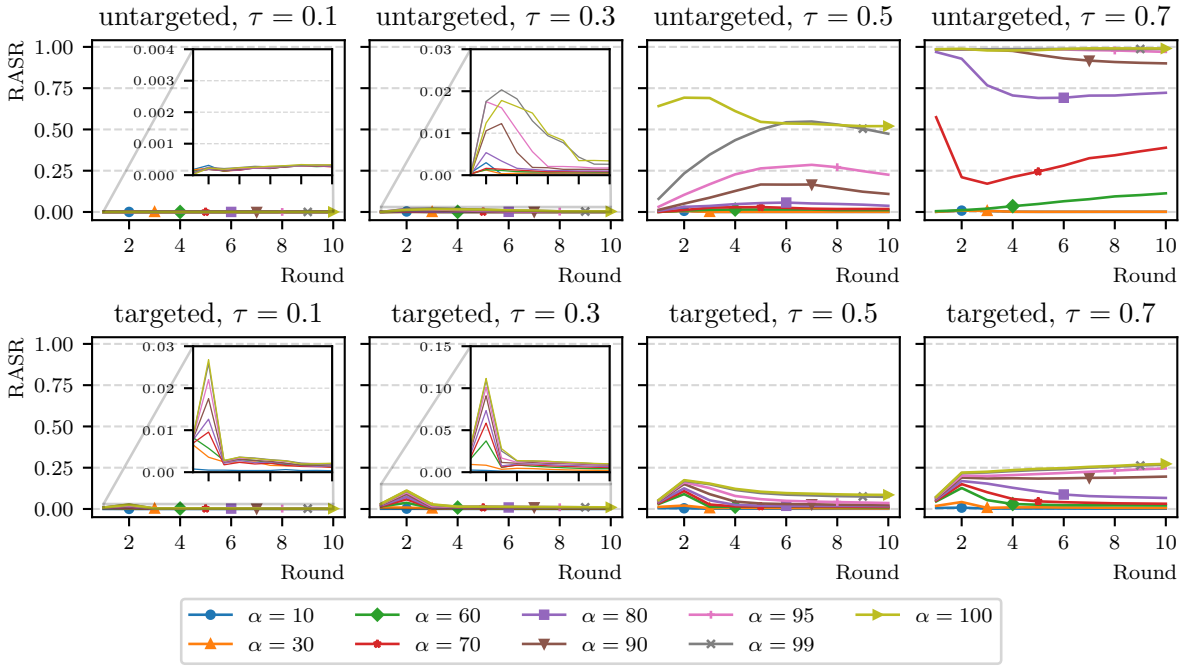
The effectiveness of label-flipping attacks is directly related the overall quantity of poisoned data. However, this relationship is not linear, and there is a critical threshold when α is below 1.0. FL suffers from same caveat as numerous other distributed systems, where the majority of participants must be honest to ensure the system's security. *However, even in its default configuration and without any defense mechanisms, multiple attackers are necessary to impact the global model's accuracy.*

5.3.5 Similarity as a Defense Mechanism

A significant amount of literature has been dedicated to the development of defense mechanisms against poisoning attacks. One of the most represented strategies is the use of similarity metrics to detect poisoned contributions [ALL21; Cao+22; FYB20; Ngu+22]. This relies on assumption that the attackers' model updates are statistically different from those of benign participants, and that a profile of either of those can be established. For instance, Tolpegin *et al.* [Tol+20] propose to use PCA to measure the distance between



(a) CIC-CSE-IDS2018



(b) UNSW-NB15

Figure 5.10 – Evolution of the RASR of poisoning attacks over time, depending on the local poisoning rate (α), the proportion of attackers (τ), and the type of attack. The x-axis represents the number of rounds. The value for targeted attacks is the mean of the targets, from which we exclude the under-represented and ineffective ones (see Figure 5.9): "Infiltration" and "Injection" in `cicids`, and "Analysis", "Backdoor", "Shellcode", and "Worms" in `nb15`. The FL round is used as the time unit.

Table 5.8 – Experiment parameters for RQ3-5.

Is gradient similarity enough to detect label-flipping attacks?	
distribution	10-0, 9-1, 5-5
scenario	continuous-100
target	untargeted
partitioner	iid_drop_1, kmeans_drop_1, iid_drop_2, kmeans_drop_2, iid_keep_1, iid_full, kmeans_full, kmeans_keep_1
dataset	cicids, nb15

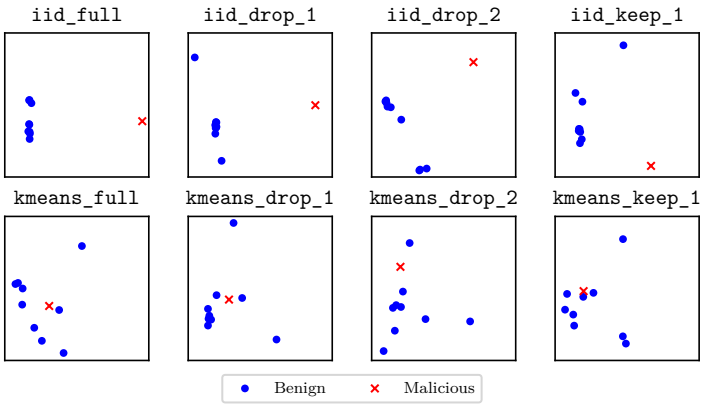
model updates in the projected space. To assess the effectiveness of this strategy for CIDSs, we explore the impact of different partitioning schemes (*c.f.* Section 5.2.4.1) on the PCA analysis. Table 5.8 summarizes the parameters used for this experiment.

Figures 5.11 and 5.12 present 2D projections of the participants' gradients using PCA for different partitioning schemes. Each point represents a client's contribution to the global model's update at the 10^{th} round. Since `FedAvg` aggregates models by default, we compute the gradients g_i^r as the difference between the participant's model at round r and the last global model, *i.e.*,

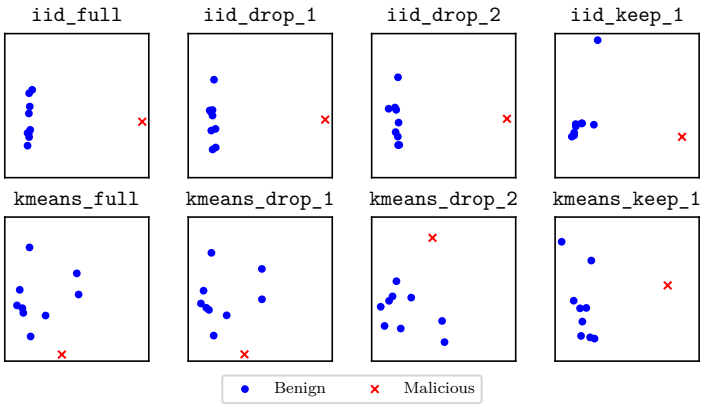
$$g_i^r = \bar{w}^{r-1} - w_i^r. \tag{5.6}$$

This allows us to compare the participants' *direction*, rather than their *position* in the model's parameter space.

The results on the `iid_full` setting are consistent with the literature: the benign participants' contributions are tightly clustered, and the attacker's contributions are easily identifiable as outliers [Tol+20]. However, the more heterogeneous partitioning schemes make it less clear, especially with a single attacker. For instance, the `kmeans_keep_1` partitioning scheme on `cicids` (Figure 5.11a) displays two outliers that are approximately as far from the benign participants, one of which is the attacker. In the `kmeans_*` schemes, the attacker is sometimes indistinguishable from the benign participants. Colluding attackers are more easily identifiable, as they are usually grouped together in the less heterogeneous settings. This quality disappears with the `kmeans_*` schemes, especially with `cicids` (Figure 5.12a). The two datasets exhibit slightly different behaviors, with attackers being more easily identifiable on `nb15` than on `cicids`, depending on the partitioning scheme.



(a) CIC-CSE-IDS2018

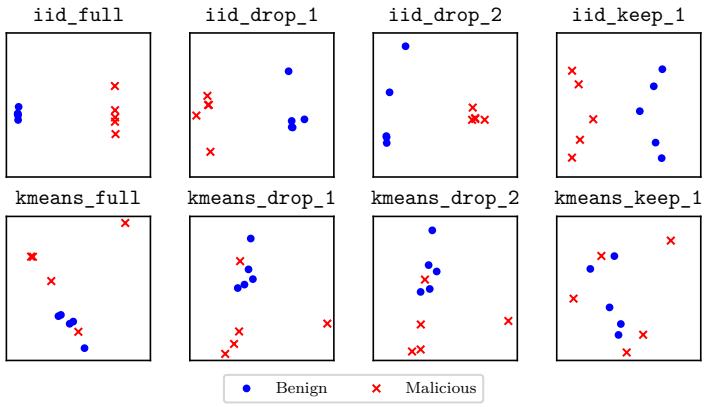


(b) UNSW-NB15

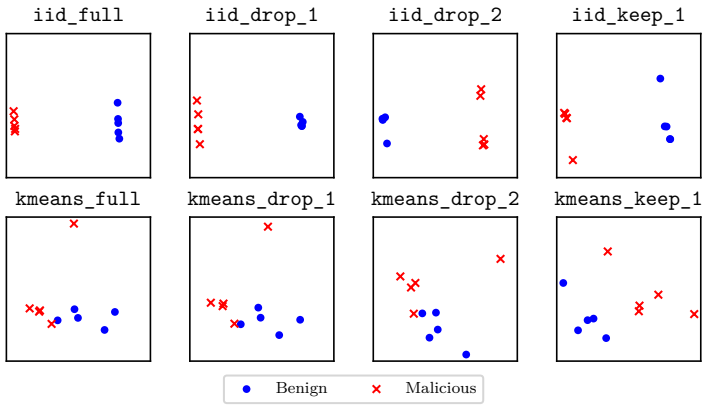
Figure 5.11 – 2D projection of gradients using PCA for different partitioning schemes with a single attacker. Results for the 10$^{\text{th}}$ round with `seed=1128`, τ=0.1, and α=1.0.

Answering RQ3-5.

The PCA analysis provides an easy and visual way to detect attackers, as long as the data distribution is homogeneous enough. However, this defense mechanism is particularly challenged in more heterogeneous settings. Colluding attackers are more easily identifiable as they will usually form a cluster of their own, highlighting the relevance of mitigation strategies such as `FoolsGold` [FYB20] and `CONTRA` [ALL21] which detect colluding attackers in heterogeneous environments using their similarity. Overall, similarity-based defense mechanisms are effective in detecting attackers in homogeneous environments, *but their effectiveness decreases with the heterogeneity of the data distribution.*



(a) CIC-CSE-IDS2018



(b) UNSW-NB15

Figure 5.12 – 2D projection of gradients using PCA for different partitioning schemes with colluding attackers. Results for the 10^{th} round with `seed=1128`, τ=0.5, and α=1.0.

5.4 Related Work

The literature on the impact of poisoning attacks on FL is extensive [Bha+19; NM22; Sun+22; Tol+20], and provides insights on the behavior of poisoning attacks on generic Machine Learning (ML) tasks, such as image classification or natural language processing. Nuding and Mayer [NM22] focus specifically on backdoor attacks, and emphasize on the importance of the choice of the trigger pattern. Fang *et al.* [Fan+20b] and Sun, Cong, *et al.* [Sun+22] rather study model-poisoning attacks. While often more effective than data-poisoning attacks, they are more complex to implement, as they require access to the uploaded models and knowledge of their functioning. The work of Tolpegin *et al.* [Tol+20] is the closest to ours, as it focuses only on label-flipping attacks. Among the most notable outcomes, the authors exhibit that targeted attacks are especially effective, having small to no impact outside the targeted class. The specificities of the IDS use case, and notably the overlap between classes, slightly contradict these conclusions.

In the context of IDSs, the literature on the impact of poisoning attacks on FL is scarcer. Zhang, Zhang, Zhang, *et al.* [Zha+22a] provide a systematic analysis of clean-

label data-poisoning attacks, where they use Generative Adversarial Networkss (GANs) to generate poisoned samples. Other works discuss clean-label attacks to a lesser extent [Ngu+20; Vy+21]. Meanwhile, Merzouk *et al.* [Mer+23] provide a comprehensive analysis on data-poisoning attacks in FL for IDSs, but focus only on trigger backdoor attacks. ML backdoors work by manipulating samples to associate a specific trigger pattern with a given class so that the model misclassifies samples containing the trigger pattern. Compared with the results of Section 5.3.3, these attacks appear to be more effective at permanently introducing IDS backdoors. Finally, Yang, He, *et al.* [Yan+23] discuss the specific aspects of label-flipping attacks in the context of FL for IDSs, using two different datasets, NSL-KDD [Tav+09] and UNSW-NB15 [MS15]. However, they only implement label-flipping as a random selection of malicious samples to be flipped, which makes the results less comparable.

5.5 Conclusion and Takeaways

The literature on the impact of poisoning attacks on FL that specifically cover intrusion detection use cases is scarce, and in it, label-flipping attacks have been overlooked. This chapter fills this gap by providing a comprehensive analysis of the impact of label-flipping attacks on FL for IDSs. We evaluated the impact of untargeted and targeted label-flipping attacks on the performance of FL models trained on CSE-CIC-IDS2018 and UNSW-NB15 using a standardized feature set to enable the extension of this work.

Our results highlight that (i) label-flipping attacks can have a significant impact on the performance of FL models, especially targeted ones; (ii) the ASR is closely related to the number of flipped samples overall, which can be approximated in IID settings by the product of DPR (α) and MPR (τ); (iii) targeted label-flipping attacks strive on well-detected targets, but can be significantly mitigated by the model's generalization capabilities; (iv) mitigation strategies must be adapted to the use case specificities (*e.g.*, constrained environments); (v) gradient similarity *can* be used to detect label-flipping attacks, but its effectiveness is challenged in heterogeneous settings.

Yet, we hope that this work will inspire and fuel further research, as there are still many open questions to address. First, our results can easily be extended with more granular experiments and testing different attack combinations. On the other hand, while the comparison with existing works seems to corroborate our results, this study calls to be extended to other datasets, feature sets, model architectures, and FL aggregation strategies. Finally, the provided evaluation framework can be used to evaluate the efficiency of existing countermeasures, or to develop new ones. We strongly believe that this work is a first step towards better evaluation of FL models and aggregation strategies in the context of intrusion detection. In the next chapter, we will build on this work's findings in heterogeneous settings and propose a novel approach to detect data-related Byzantine

faults in heterogeneous FIDS environments.

Chapter 6

Fighting Byzantine Contributions in Heterogeneous Settings

Contents

6.1 Introduction

In the previous chapters, we identified and studied two major challenges that currently hinder the adoption and deployment of Federated Intrusion Detection Systems (FIDSs): (1) the heterogeneity of the data sources, notably in Cross-Silo Federated Learning (CS-FL) settings; and (2) the susceptibility of FIDSs to adversarial attacks. More generally, because collaborative systems are inherently sensitive to input quality, any form of Byzantine failure should be considered. While we focus specifically on data-related failures in the context of this thesis, Byzantine faults can also encompass other types of failures, such as crashes, arbitrary behavior, or communication issues. This applies whether the participants are honest but use faulty data, or actively malicious. In this heterogeneous context, it is particularly challenging to distinguish a faulty or malicious contribution from a legitimate one originating from a different type of infrastructure.

Approaches that assess model quality [PB23] or mitigate poisoning [Bla+17; Cao+22] in homogeneous distributions typically compare or evaluate a model using a single source of truth. Building such a single source of truth, however, is inadequate in heterogeneous contexts due to the differences between participants. Assuming that all contributions are therefore different, some approaches detect colluding attackers based on their similarity [ALL21; FYB20]. Nevertheless, these approaches fail to detect isolated attackers.

In this chapter, we present RADAR, an architecture for CS-FL guarantying high-quality model aggregation, regardless of the data homogeneity. RADAR relies on three main ingredients: *i)* a modified Federated Learning (FL) workflow, where each participant uses its local dataset to evaluate the other participants' models, between the training and aggregation steps; *ii)* a clustering algorithm leveraging the participants' perceived similarity to aggregate group-specific global models; and *iii)* a reputation system that weights the participants' contributions based on their past interactions.

We evaluate the performance of RADAR in a realistic Collaborative Intrusion Detection System (CIDS) use case, using four network flow datasets with standardized features, representing different environments, and model various Byzantine behavior using label-flipping. We also compare our approach to existing strategies [FYB20; McM+17], and conclude that RADAR can detect Byzantines contributions under most scenarios, from noisy labels to colluding poisoning attacks.

The content of this chapter is based on our work published in IEEE International Symposium on Reliable Distributed Systems (SRDS) [Léo+24], which results from a collaboration with Pierre-Marie Lechevalier, another Ph.D. student at IMT Atlantique. It is organized as follows. Section 6.2 introduces the reader to the problem of model quality assessment in CS-FL and the necessary background. Section 6.3 reviews the related work, before we dive in RADAR's architecture in Section 6.4. Sections 6.5 and 6.6 present the experimental setup and results, and we discuss our findings in Section 6.7. Finally, Section 6.8 concludes this chapter.

Contributions of this chapter

- RADAR, an architectural framework to protect FL strategies using clustering and reputation-aware aggregation, validated by extensive evaluation against relevant baselines;
- a demonstration that evaluation metrics (such as accuracy, F1-score, or loss) can be used to effectively assess similarity between FL participants, and as an input to clustering and reputation algorithms;
- the confirmation that combining reputation and clustering successfully addresses the problem of contribution quality assessment in heterogeneous settings.

6.2 Problem Statement

In continuity with Chapters 4 and 5, we consider once more the use case introduced in Section 4.2 and the associated datasets. Specifically, we focus on a heterogeneous decli-

nation of this CIDS use case, where we admit that participants share similarities in their data distributions—*e.g.*, between organizations operating in the same sector or having similar network infrastructure. This setting, also mentioned in Section 4.2, is referred to as *practical* Non Independent and Identically Distributed (NIID) [Hua+21]. We also set $C = 1$, as we consider that the participants are highly available and interested in collaborating.

6.2.1 Low-quality Contributions

In FL, the quality of the global model is directly impacted by the quality of the participants' contributions. In a Intrusion Detection System (IDS) context, the poor quality of a Machine Learning (ML) model can be induced by some choices in terms of architecture, hyperparameters, or optimizer—all fixed by the server, but also by the quality of the training data. Multiple factors can affect the quality of local training data [Jai+20], such as: (1) *Label noise*—samples associated with the wrong labels; (2) *Class imbalance*—differences in terms of class representation in the dataset; or (3) *Data heterogeneity*—the variations between samples of the same class.

Similar to existing works on data-quality [Den+21; Den+22], we focus on label noise, which can have significant consequences on the global model's performance, depending on the proportion of mislabeled samples. In a CIDS, label noise can unknowingly be introduced by the participants, either due to misconfigurations or to the presence of compromised devices. We consider two types of label noise: *missed intrusions* and *misclassification*.

a) *Missed intrusions* occur when a malicious sample is mislabeled as benign, leading to a false negative. Participants in CIDSs label the attacks they are aware of, but some might have been unnoticed.

b) A *misclassification* is the random mislabeling of a sample. This can be due to a lack of knowledge or to a misconfiguration.

Such participants are referred to as *honest-but-neglectful*. Because these errors are assumed to be unintentional, the proportion of *misclassified* samples is expected to be low. However, the concept of *missed intrusions* implies that the participants are not aware of an entire attack, which can represent a significant proportion of their dataset.

6.2.2 Data Poisoning Attacks

In addition to accidental low-quality contributions, some participants might deliberately upload model updates that would negatively impact the performance of the global model. Specifically, we consider the same attack model as detailed in Chapter 5, and focus on label-flipping attacks. The model can be summarized as follows:

Attackers' Knowledge. Attackers are *gray-box* adversaries, meaning that they have access to the same information as the other participants; *e.g.*, the last global models, the hyperparameters, or the optimizer.

Attackers' Objective. With targeted poisoning, attackers aim at making a specific type of attack invisible to the Network-based Intrusion Detection System (NIDS). Conversely, with untargeted attacks, they seek to jeopardize the NIDS performance by maximizing the number of misclassifications.

Attackers' Capabilities. Attackers can flip the labels of an arbitrary proportion of their dataset, referred to as the Data Poisoning Rate (DPR) and denoted α. They can act alone or in collusion with other by applying the same strategy. The proportion of attackers in the system is described by the Model Poisoning Rate (MPR) and denoted β.

Because we do not make a priori assumptions on the whether the participants are malicious or not in this contribution, we also refer to the DPR as the *noisiness* of a participant. The MPR, on the other hand, almost exclusively describes attackers, as it is unlikely for the same Byzantine fault to occur in multiple participants simultaneously.

6.2.3 Problem Formalization

Based on the previous assumptions, we consider that participants might upload model updates that would negatively impact the performance of the global model, deliberately or not. Multiple forms of such actors can exist: external actors altering legitimate clients' data (*i.e. compromised*), clients whose local training sets are of poor quality (*i.e. honest-but-neglectful*), or clients modifying their own local data on purpose (*i.e. malicious*). We refer to them as *Byzantine participants* or simply *Byzantines* in the remaining of this paper.

We further consider that the server can be trusted to perform the aggregation faithfully, and that FL guaranties the confidentiality of the local datasets. Attacking the server is out of the scope of this contribution. Consequently, we aim at weighting or discarding the participants' contributions based on their quality to guaranty the performance of the aggregated model.

Problem 6.1: Quality Assessment in Heterogeneous Settings

For n participants p_i and their local datasets d_i of unknown similarity, each participant uploads a model update w_i^r at each round r. Given $P = \{p_1, p_2, \ldots, p_n\}$ and $W = \{w_1^r, w_2^r, \ldots, w_n^r\}$, how can one assess the quality of each participant's contribution w_i^r without making assumptions on the data distribution across the datasets d_i?

6.3 Background and Related Work

The reliability of a submitted local model can be assessed in several ways, whether it is used to detect *honest-but-neglectful* or explicitly *malicious* participants. In this section, we review the existing literature on model quality assessment in FL and the related work on Byzantine-robust FL. We review the existing approaches to detect and mitigate the impact of Byzantine contributions, and discuss the limitations of these methods in heterogeneous settings. We also review the existing works on reputation systems in FL and the use of clustering to improve the aggregation of local models.

6.3.1 Byzantine-resilient Federated Learning

Some approaches use evaluation to validate submitted models against a centralized dataset [Cao+22], or against randomly selected distributed datasets [PB23] if they are representative of each other—which is the case with Independent and Identically Distributed (IID) data partitioning. Given IID settings, submitted models can also be compared to each other [Bla+17; Cao+22; Ngu+22] or with a reference model [XTL21; Zho+22], using distance metrics. Among these, `FLAME` [Ngu+22] stands out, as it leverages multiple complementary methods to stop malicious participants: clustering to identify *multiple* groups of attackers, norm-clipping to mitigate gradient boosting attacks, and adaptive noising to lessen the impact of outliers. Yet, because it works under the assumption that the biggest cluster represents benign participants and that attackers cannot exceed 50% of the population, `FLAME` *de facto* falters against a majority of malicious clients. Furthermore, while the paper demonstrates that it can resist to low proportions of *NIID* participants, it still aims at delivering one common global model, thus failing to address the more skewed NIID cases, where leveraging multiple sub-federations might be necessary.

The assumption of IID data rarely holds in FL, even though its properties facilitate the detection of Byzantine participants. Indeed, given NIID settings, You *et al.* [You+22] show most of these mitigation strategies are inefficient. These methods rely on a single source of truth that may be known beforehand [Cao+22], or elected among participants [Bla+17]. However, by definition, this single source of truth does not exist in NIID datasets. To circumvent this issue, `FoolsGold` [FYB20] and `CONTRA` [ALL21] assume that sybils share a common goal, and thus produce similar model updates, allowing to distinguish them from benign NIID participants that present dissimilar contributions. Similar participants are classified as sybils using the cosine similarity between gradient updates, and their weight is reduced in the final aggregation. However, while this mitigation strategy works when multiple attackers collaborate, it fails at identifying lone attackers. These approaches are also well suited for *pathological NIID* scenarios, where all participants are significantly different. In *practical NIID* settings, legitimate communities of similar participants can exist. Those legitimate participants would be falsely identified as sybils.

Finally, Zhao, Hu, *et al.* [Zha+20c] take a different approach and rely on client-side evaluation. Local models are aggregated into multiple sub models, which are then randomly attributed to multiple clients for efficiency validation. To also address NIID datasets, clients self-report the labels on which they have enough data to conduct an evaluation. While this self-reporting limits the network and client resources consumption, abusive self-reporting is possible. Nevertheless, directly leveraging the participant datasets for evaluation removes the need for a single exhaustive source of truth. Resource consumption is also less of an issue in cross-silo use cases: they often imply fewer participants, with more data and dedicated resources.

6.3.2 Clustered Federated Learning

NIID data can also be regarded as heterogeneous data distributions $\mathcal{P}_k$ that are regrouped together, where $\mathcal{P}_k$ is the distribution of the dataset d_k. Following this idea, some works [BFA20; Ouy+22; Ye+23] try to group participants sharing similarities. The purpose of this approach is twofold. First, from a performance perspective, NIID settings slow down convergence. Even if a global minimum is reached, the model might not be optimal for all participants [Kai+21; Ouy+22]. In addition, considering outliers as poisoned models [Per+20], one can eliminate thier in the aggregation process.

Since the effective number of clusters is unknown, hierarchical clustering is a common way to create appropriate clusters [BFA20; Ye+23]. Specifically, Ye *et al.* [Ye+23] use the cosine similarity of local models to successfully group participants in more homogeneous subgroups. However, as this approach doesn't aim to address Byzantines, it does not consider that some malicious participants might aim to be grouped with benign ones to poison the cluster's model. Another approach for finding the appropriate number of clusters is dynamic *split-and-merge* clustering [Che+21], where the number of clusters is adjusted depending on the distance between the participants' in each cluster. Finally, Ouyang *et al.* [Ouy+22] propose a clustering algorithm relying on K-means and spectral relaxation to group participants without prior knowledge of the number of clusters. Contrary to the most of the existing works, they do not use metrics that rely on vector representations of the models (such as cosine similarity, L2 norm, or scalar products). Rather, they leverage the Kullback-Leibler Divergence (KLD) to compare the models' probability distributions, which do not require the models to rely on a convex loss function.

6.3.3 Reputation systems for Federated Learning

In collaborative applications, reputation systems preemptively assess the ability of participants to perform a task and the quality of its result, based on past interactions. Definition 6.1 provides a formal definition of reputation systems. In the context of FL, they usually have three main applications: (i) client selection; (ii) model weighting and

aggregation; and (iii) tracking contribution quality over time.

Definition 6.1: Reputation Systems

A reputation system collects, distributes, and aggregates feedback about participants past behavior. [...] To operate effectively, reputation systems require at least three properties:

- Long-lived entities that inspire an expectation of future interaction;
- Capture and distribution of feedback about current interactions (such information must be visible in the future); and
- Use of feedback to guide trust decisions.

– Resnick *et al.* [Res+00]

The first application, client selection, is used to determine which participants should be included in the training process of the next round [ALL21; Kan+20; Son+22; Tan+22]. This is particularly useful in scenarios with constrained resources [Son+22] and in hybrid architectures (see Figure 2.4b) where servers can exchange reputation information about their users [Kan+20]. `CONTRA` [ALL21] provides an example of such a reputation system for client selection. By progressively penalizing the participants that propose models similar to each others, and that are thus suspected of being *sybils* (see Section 6.2 and **??**), it leaves room for participants issuing dissimilar models to be selected more often. We detail in **??** the limits of these types of approaches in practical NIID settings.

The second main application is to weight local models during the aggregation process [Wan+22; WK21]: the higher the reputation, the heavier the local model contributes to the aggregated model. Some will even go so far as to discard contributions when the author's reputation is too low. Karimireddy, He, and Jaggi [KHJ21] underline the importance of historical record in robust aggregation: malicious incremental changes can be small enough to be undetected in a single round but still eventually add up enough to poison the global model over the course of multiple round. Reputation system's ability to track clients' contributions over time [Kan+20; WK21] can be used as a countermeasure to these attacks.

Finally, note that the literature on reputation systems sometimes distinguishes between *reputation* and *trust* systems [Che+11; ZY15]. One of the main differences is the use of indirect feedbacks in reputation systems, wheras trust systems rely on direct evaluation an objective metrics. Based on this distinction, the reputation is the global perception of a one's trustworthiness in the system, based on the feedback of others [Che+11]. To the best of our knowledge, no work has yet been published in the context of FL that suit this definition.

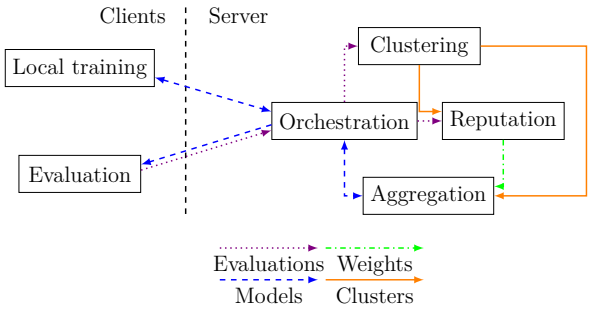


Figure 6.1 – *Architecture overview.*

6.4 Architecture

This section details RADAR's architecture. It is divided into three main components: (*i*) our cross-evaluation scheme that provides local feedbacks on each participant's contributions (Section 6.4.1), (*ii*) a similarity-based clustering algorithm that groups participants based on evaluations (Section 6.4.2), and (*iii*) a reputation system that assesses participants' trustworthiness based on their past contributions (Section 6.4.3). Figure 6.1 provides an overview of RADAR.

6.4.1 Assessing Contributions with Cross-Evaluation

As highlighted in Section 6.3, most related works on poisoning mitigation in FL rely on server-side models comparison [ALL21; FYB20]. They measure distance between the parameters (for Deep Neural Networks (DNNs), n-dimensional arrays containing the weights and biases of each neuron) using metrics such as cosine similarity [FYB20] or Euclidean distance [Ma+22]. However, models that are statistically further from others are not automatically of poor quality. To cope with this limitation, as well as the absence of source of truth, we propose to rely on client-side evaluation [Zha+20c]. The results of this evaluation can then be used by the server to either discard or weight contributions. RADAR's workflow thus differs from typical approaches by adding an intermediate step for evaluating parameters:

1. *client fitting*—The server sends clients training instructions and initial parameters, *i.e.* randoms values for the first round. For subsequent rounds, the initial parameters of each client are initialized as the aggregated model (denoted $\bar{w}_i^{r-1}$) of the corresponding cluster, using the results of Step 3. at round $r-1$. Each client trains its own model using the provided hyperparameters, and the initial parameters as a starting point before uploading their parameters w_i^r to the server.
2. *cross-evaluation*—The server serializes all client parameters in a single list that is sent to every client. Each client then locally evaluates each received model using its validation set, generating a predefined set of metrics such as loss, accuracy, or

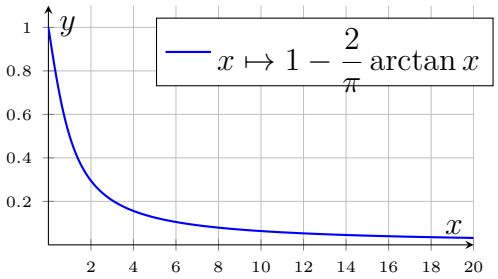


Figure 6.2 – Loss normalization function.

F1-score. The metrics of all clients are then gathered server-side.

3. *parameter aggregation*—The server partitions clients into a set of clusters $\mathscr{C}$ based on the evaluations gathered in Step 2. For each cluster $C_k \in \mathscr{C}$, the server computes the new model $\bar{w}_k^r = \sum_{i|p_i \in C_k^r} \rho_i^r w_i^r$, where the weight ρ_i^r is given by the reputation system for the participant p_i at round r.

The cross-evaluation step generates an evaluation matrix that is used twice in the architecture. Since this matrix is not symmetric, the vector of *issued evaluations* $E_{[i,*]}^r$ is used for clustering, while both the *received evaluations* vector $E_{[*,j]}^r$ and the *issued evaluations* vector $E_{[i,*]}^r$ are used in the reputation system. Algorithm 6.1 details the proposed workflow.

While most of the metrics for ML (see Section 2.2.3) are strictly expressed between $[0, 1]$, the loss value is expressed in $[0, \infty[$, and is inverted when compared to the accuracy, for instance. The lower the loss, the better the model parameters w_j^r of a participant p_j fit the dataset d_i of another participant p_i. This poses an issue for harmonizing metrics before using them in a clustering or reputation algorithm. Thus, to project the loss value into a comparable space, we need to use a continuous, strictly decreasing function mapping $\mathbb{R}^+$ to $[0, 1]$. We choose to use $x \mapsto 1 - \frac{2}{\pi} \arctan x$ (see Figure 6.2), as it emphasizes the lower part of the spectrum, where the differences between model losses are concentrated.

6.4.2 Fighting Heterogeneity with Clustering

The clustering algorithm seeks to gather similar participants together in more homogeneous sub-federations when appropriate. Nguyen, Rieger, Chen, *et al.* [Ngu+22] and Ye *et al.* [Ye+23] both measure participants' similarity by comparing the distance between model updates. This is biased, as models that are statically different might still produce relevant results. RADAR addresses this issue by defining similarity as the distance between participants emitted evaluations. Indeed, since all participants evaluate the same models, the variation in evaluation results reflects a difference in the evaluation datasets. Therefore, participants having similar datasets should issue similar evaluations.

We note $\delta_{i,j}^r$ the distance between the evaluations of p_i and p_j at round r. $\delta_{i,j}^r$ is de-

Algorithm 6.1 RADAR. R is the number of rounds, β the local batch size, η the learning rate, ε the number of epochs, and $\mathcal{L}$ a loss function. ω and Ω represent the model and the set of models that are passed to the clients, respectively. We highlight in `blue` the elements that differ from the standard FL workflow (see Algorithm 2.1 in Section 2.4).

Require: P

with $r \leftarrow 0$ **do**
  $\mathscr{C}^r \leftarrow \{P\}$
  $\overline{W}^r \leftarrow (\textsc{Random}(\,))$

for $r \leftarrow 1, \ldots, R$ **do**
  ▷ *Step (1): model training* ◁
  for all $p_i \in P$ **in parallel do**
    $k \leftarrow \textsc{GetCluster}(p_i, \mathscr{C}^r)$
    $w_i^r \leftarrow \textsc{ClientFit}(p_i, \overline{w}_k^r)$
  $W^r \leftarrow (w_i^r)_{i \in n[\![1,n]\!]}$
  ▷ *Step (2): cross-evaluation* ◁
  for all $p_i \in P$ **in parallel do**
    $(e_{i,j}^r) \leftarrow \textsc{ClientEvaluate}(p_i, W^r)$
  $E_{[i,j]}^r = [e_{i,j}^r]_{i,j \in [\![1,n]\!]}$
  ▷ *Step (3): parameters aggregation* ◁
  $\mathscr{C}^r \leftarrow \textsc{ComputeClusters}(E^r)$ ▷ *See: Section 6.4.2*
  for all $C_k^r \in \mathscr{C}^r$ **do**
    $(\rho_i^r) \leftarrow \textsc{ComputeReput}(E^r, \mathscr{C}^r)$ ▷ *See: Section 6.4.3*
    $\overline{W}^r \leftarrow \dfrac{1}{|C_k^r|} \displaystyle\sum_{i=0}^{|C_k^r|} w_i^r$

function $\textsc{ClientFit}(p, \omega)$
  for $i \leftarrow 1, \ldots, \varepsilon$ **do**
    for all $b \in \textsc{Split}(d_i, \beta)$ **do**
      $\omega \leftarrow \omega - \eta \nabla \mathcal{L}(\omega, b)$
  return ω

function $\textsc{ClientEvaluate}(p, \Omega)$ ▷ *On client.*
  for all $\omega_j \in \Omega$ **do**
    $e_{i,j}^r \leftarrow \textsc{Eval}(\omega, d_i)$
  return $(e_{i,j}^r)_{i,j \in [\![1,n]\!]}$

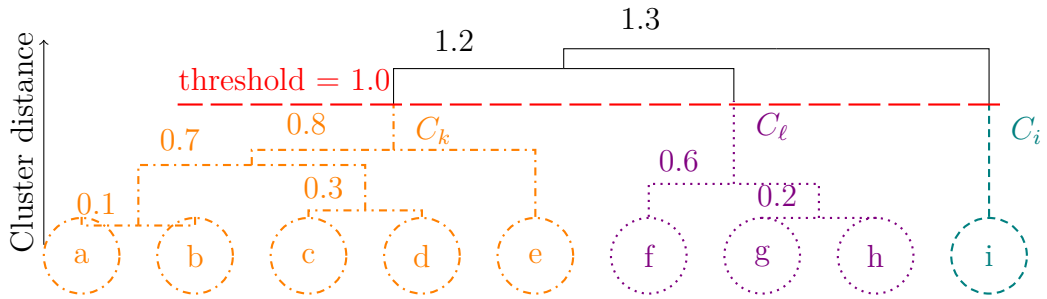


Figure 6.3 – Hierarchical clustering process.

fined as the cosine similarity between p_i and p_j issued evaluation vectors $E^r_{[i,*]}$ and $E^r_{[j,*]}$, or $\delta(E^r_{[i,*]}, E^r_{[j,*]})$. We then iteratively group similar participants into different clusters, leveraging hierarchical clustering. Initially, each participant is assigned to a different cluster. Then, each closest pair of clusters is merged, thus reducing the number of clusters. The process is repeated until the distance between the two closest clusters exceeds a given threshold. Figure 6.3 illustrates this process.

While hierarchical clustering does not require the number of clusters as an input, choosing the right threshold can be challenging. Contrarily to Ye *et al.* [Ye+23] who manually adjust this parameter on a per-dataset basis, RADAR leverages a dynamic threshold based on the mean inter-distance $\overline{\Delta^r}$ between the clusters at round r. This threshold Θ is expressed as:

$$\Theta = \beta\overline{\Delta^r} = \frac{\beta}{|\mathscr{C}^r|(|\mathscr{C}^r|-1)} \sum_{\substack{k,\ell \in \mathscr{C}^r, \\ k \neq l}} \Delta^r_{k,\ell} \tag{6.1}$$

where β is a tunable hyperparameter, and $\Delta^r_{k,\ell}$ the distance between two clusters C^r_k and C^r_ℓ, defined as the distance between their centroids: $\delta(\mu^r_k, \mu^r_\ell)$. The centroid μ^r_k of a cluster C^r_k is the average of the issued evaluations from its participants at round r, *i.e.*, we have $\mu^r_k = \frac{1}{|C^r_k|} \sum_{i \in C^r_k} E^r_{[i,*]}$.

Based on the results of the clustering, the server can then aggregate the models of each cluster C^r_k separately, using the reputation system described in Section 6.4.3. Consequently, the server maintains as many global models $\overline{w}^r_k$ as there are clusters at each round. Note that this is another difference with FLAME [Ngu+22], which only produces a single common model for every participant.

6.4.3 Ensuring Quality Contributions with Reputation

The reputation system centrally computes the weights $\rho^r_i, \forall p_i \in C^r_k$ used in the aggregation of each cluster model $\overline{w}^r_k$ at round r (see Section 6.4.1). Given the existence of methods for common tasks, such as contribution filtering, RADAR models trust using a multivalued Dirichlet probability distribution [Fun+11]. However, the evaluations $E^r_{[*,i]}$ received by a participant p_i are continuous over $[0, 1]$, and thus need to be discretized into

a set of q possible values $\varepsilon = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_q\}$.

A Dirichlet distribution on the outcome of an unknown event (*i.e.*, the mean of the received evaluation $\frac{1}{n} \sum_{e^r_{i,j} \in E^r_{[*,j]}} e^r_{i,j}$) is usually based on the combination of an initial belief vector and a series of cumulative observations [Fun+11]. As a complete cross evaluation is already available at the first round, RADAR does not require an initial belief vector to bootstrap reputation.

Following the notation used by Fung, Zhang, *et al.* [Fun+11], we note $\vec{\gamma}^r = \{\gamma^r_1, \gamma^r_2, \ldots, \gamma^r_q\}$ the cumulative evaluations received by p_i: $\gamma^r_2 = 3$ means that three evaluations in $E^r_{[*,j]}$ had values bounded by $\left[\frac{1}{q}, \frac{2}{q}\right[$. We then note $\vec{\mathbb{P}} = \langle \mathbb{P}\{\epsilon_1\}, \mathbb{P}\{\epsilon_2\}, \ldots, \mathbb{P}\{\epsilon_q\} \rangle$ the probability distribution vector for the received evaluation of a participant, where $\sum_{s=1}^{q} \mathbb{P}\{\epsilon_s\} = 1$. Leveraging the cumulative evaluations $\vec{\gamma}^r$, the probability $\mathbb{P}\{\epsilon_s | \vec{\gamma}^r\}$ is given by $\mathbb{P}\{\epsilon_s | \vec{\gamma}^r\} = \gamma_s / \sum_{m=1}^{q} \gamma_m$.

The system further needs to limit the ability of potential malicious participants to manipulate their evaluations, either by badmouthing another participant, or by artificially raising their own ratings. Consequently, the evaluations issued by a participant $p_i \in C^r_k$ are weighted according to their similarity with other cluster members' [XL04] as $e'^r_{i,j} = e^r_{i,j} sim(E^r_{[i,*]}, E^r_{[C^r_k,*]})$, where the similarity is defined as:

$$sim(E^r_{[i,*]}, E^r_{[C^r_k,*]}) = 1 - \sqrt{\frac{\sum_{j=1}^{n} \left(e^r_{i,j} - \sum_{i \in C^r_k} \frac{e^r_{i,j}}{|C^r_k|} \right)^2}{|P|}}. \tag{6.2}$$

To prevent attacks phased over multiple rounds, while preventing past mistakes from permanently impacting a participant, we use an exponential decay as forgetting factor, noted $\lambda \in [0, 1]$. The reputation ψ^r_i of a participant p_i at round r based on the prior knowledge γ^r_i of this participant is given by Equation (6.3). Note that a small λ gives more importance to recent evaluations: $\lambda = 0$ only considers the last round while $\lambda = 1$, considers all round with equal weight. Based on ψ^r_i, the weight ρ^r_i of w^r_i for aggregation in $\overline{w}^r_k$ (see Step 3. in Section 6.4.1) is given by Equation (6.4).

$$\psi^r_i = \sum_{\kappa=1}^{r} \lambda^{r-\kappa} \gamma^\kappa_i \tag{6.3}$$

$$\rho^r_i = \frac{\psi^r_i}{\sum_j^{|C^r_k|} \psi^r_j} \tag{6.4}$$

As such, the weight ρ^r_i of p_i will be proportional to its reputation, and therefore the evaluations it received over time. The attackers' evaluations only vary on the subset of samples that are impacted. Consequently, the differences between their reputation scores and those of legitimate participants can be relatively small, despite remaining

Table 6.1 – *Hyperparameters.* The model's configuration is taken from the work of Popoola, Gui, *et al.* [Pop+21b], while the parameters for RADAR's architecture have been selected empirically.

Model hyperparameters	
Learning rate	0.0001
Batch size	512
Hidden layers activation	ReLU
Output layer activation	Sigmoid
# Input features	49
# Hidden layers	2
# Neurons (hidden layers)	128
Optimization algorithm	Adam
Loss function	Log loss
Number of local epochs	10

Clustering hyperparameters	
Distance metric	Cosine similarity
Threshold factor β	0.25
Cross-eval. metric	F1-score

Reputation hyperparameters	
Number of classes	10000
History parameter λ	0.3
Cross-eval. metric	F1-score
Normal distribution σ	0.0005

meaningful. We apply a sigmoid function to convert these scores to aggregation weighs and accentuate this difference. This sigmoid function is the normal distribution cumulative density function adjusted with the σ parameter.

6.5 Experimental Setup

We evaluate RADAR and any selected baseline on a set of heterogeneous intrusion detection datasets [SLP22] with various attack scenarios (see Section 6.5.2). We implement the described use case (Section 4.2) and threat model (**??**) as a set of experiments using the FL framework Flower [Beu+20], with Nix [Dol06] and Poetry [Eus18] to reproducibly manage dependencies. The hyperparameters used in our setup are detailed in Table 6.1. The code for all experiments can be found online [1], with configuration and seeds for each considered baseline and evaluation scenario. We also provide lock files to enable anyone to reuse the same software versions as in this chapter.

6.5.1 Datasets and local algorithm

In continuity with the previous chapters, we implement our CIDS use case using the datasets introduced in Section 4.2 and the standard feature set for flow-based NIDSs proposed by Sarhan, Layeghy, and Portmann [SLP22]. However, to create groups of participants that share similar distributions, we use the four datasets converted to this format by the authors: UNSW-NB15 [MS15], Bot-IoT [Kor+19], ToN_IoT [Mou21], and CSE-CIC-IDS2018 [SHG18]. The uniform feature set allows evaluating FL approaches on independently generated datasets [dCar+23; Pop+21b].

1. TODO.

Table 6.2 – *Cross evaluation (F1-score) on the used datasets.* Each dataset is uniformly partitioned into a training set (80%) and an evaluation set (20%). The same partitions are kept over the entire experiment. Each model (rows) is trained on its training set during 10 epochs, and then evaluated on each test set (columns). The highest scores are highlighted in bold.

		Evaluation set			
Training set		CIC-IDS	NB15	ToN_IoT	Bot-IoT
	CIC-IDS	**0.961787**	0.002723	0.524219	0.680166
	NB15	0.108913	**0.947204**	0.009875	0.655943
	ToN_IoT	0.211792	0.419380	**0.966679**	0.081510
	Bot-IoT	0.158477	0.017188	0.703195	**0.999483**

We use the "sampled" version (1,000,000 samples per dataset) provided by the same team [LP22]. Like de Carvalho Bertoli *et al.* [dCar+23], we remove source and destination IPs and ports, as they are more representative of the testbed environment than of the traffic behavior. We then use one-hot encoding on the categorical features (both for samples and labels), and apply min-max normalization to give all features the same importance in model training.

Locally, we use a Multilayer Perceptron (MLP) with two hidden layers, following Popoola, Gui, *et al.* [Pop+21b]. We reuse the hyperparameters provided by the authors (see Table 6.1), and reproduce their results on our implementation, using the same four datasets. Their algorithm shows low performance when training the model on one dataset, and evaluating it on another, as illustrated in Table 6.2. This supports the assumptions behind the cross-evaluation proposal, where the differences between the evaluation results can be used to estimate the similarity between the local data distribution.

6.5.2 Evaluation scenarios

The threat model defined in Section 6.2.3 is implemented as a set of evaluation scenarios which model various data-quality situations. These scenarios can be summarized in three categories:

C1: `Benign`. This category actually contains one scenario which showcases a *practical NIID* situation, where participants can be grouped into 4 use cases. Each of the 4 datasets described in 6.5.1 is randomly distributed among 5 participants without overlap. We thus have a total of 20 participants with different data, but some share similarities between their data distributions.

C2: `Lone Byzantine`. The scenarios in this category differ from the `Benign` category (C1) by introducing a fault in a single participant. This fault might be due to an *honest-but-neglectful* participant that misclassified samples or missed an intrusion, or a single *malicious participant* actively trying to poison the system. We emulate

the fault by flipping the one-hot encoded label on a subset of the participant's data: given a label $\vec{y} \in \{\langle 0,1\rangle, \langle 1,0\rangle\}$, a faulty sample will be assigned to $\langle \neg\vec{y}_0, \neg\vec{y}_1\rangle$. A fault is characterized by two parameters:

(1) its *target*, *i.e.*, the classes to which the affected samples belong; and

(2) its *noisiness*, *i.e.*, the percentage (ranging from 10% to 100%) of targeted labels that are actually flipped.

If a single class is affected, the fault is *targeted*, and only the samples of this class see their label changed. We arbitrarily chose Bot-IoT and its "Reconnaissance" class as the target for the experiments. Otherwise, the fault is *untargeted*, and all classes of Bot-IoT are equally affected, including benign samples.

C3: `Colluding Byzantines.` This category encompasses scenarios resembling the `Lone Byzantine` ones (Category C2), but where the same fault is replicated on multiple participants at the same time. This corresponds to *malicious participants* in our threat model, as it is unlikely that several *honest-but-neglectful* participants commit the very same fault. The colluding attackers are a *majority* if they outnumber the benign participants whose data originate from the same dataset, and a *minority* otherwise. As we experiment attacks on the Bot-IoT dataset whose data is distributed among 5 participants, this respectively means that there are three attackers and two benigns, or two attackers and three benigns. These two sub categories are referred to as `Colluding majority` and `Colluding minority`, respectively.

We note the parameters of a fault as `<noisiness><initial_of_target>`, and use this notation to refer to scenarios hereafter. As such, a `Lone 80T` scenario means that one of the five participants coming from the Bot-IoT dataset will flip 80% of its "Reconnaissance" labels to the opposite value. `Colluding minority` $\leq$`30U` refers to all scenarios where two participants from Bot-IoT flip the labels on 30% of their entire dataset, or less.

6.5.3 Metrics

To measure the ability of `RADAR` to cluster clients correctly, we use the Rand Index. The Rand index compares two partitions by quantifying how the different element pairs are grouped in each. It is defined between 0 and 1.0, 1.0 meaning that both partitions are identical. `RADAR` already produces evaluation metrics at each round thanks to the cross-evaluation scheme, based on each participant's validation set. The same evaluation methods are thus used on a common testing set (to each initial client dataset) and aggregated to evaluate the approach. The presented results focus on the mean accuracy and miss rate of the benign participants. Finally, the Attack Success Rate (ASR) is computed over the benign participants of the affected cluster, and defined as the mean miss rate on the targeted classes of targeted attacks, and the mean of the misclassification rates (*i.e.* $1 - accuracy$) in untargeted ones.

Table 6.3 – *Rand Index between* RADAR*'s clustering and two partitions of reference, under various scenarios.* Partition **(A)** contains only benign participants grouped according to their respective dataset. Partition **(B)** contains attackers placed in a separated group in addition to benign participants.

Scenario			Partition (A)	Partition (B)
Category	*Noisiness*	*Target*		
Benign			1.00	1.00
Lone	≤100	T	1.00	0.97
Lone	≤95	U	1.00	0.97
Lone	100	U	1.00	1.00
Collud. min.	≤100	T	1.00	0.97
Collud. min.	≤90	U	1.00	0.97
Collud. min.	100	U	1.00	1.00
Collud. maj.	≤100	T	1.00	0.96
Collud. maj.	≤90	U	1.00	0.96
Collud. maj.	100	U	1.00	1.00

6.6 Experimental Results

RADAR serves multiple objectives at once: (a) maintaining high performance on *practical* NIID data, (b) correctly identifying and weighting low-quality contributions, and (c) mitigating the impact of label-flipping attacks. As a result, we select relevant baselines from the literature to evaluate each of RADAR's abilities. We use FedAvg [McM+17] (abbreviated FA) to highlight the existing issues with statistical heterogeneity, using the setup provided by Flower [Flo24]. Because RADAR can be partially assimilated as a clustered FedAvg variant, we also consider a theoretical setup where participants are clustered based on their original data distribution, and one instance of FedAvg is executed per cluster. We refer to it as *Clustered* FedAvg or FC. To highlight RADAR's ability to compare with Sybil-focused mitigation strategies, we compare it with FoolsGold [FYB20] (also designated FG). We reuse the authors' code [FYB19], and adapt it to model updates, since FoolsGold was originally implemented on FedSGD. The following sections cover these topics using the scenarios laid out in Section 6.5.2. Like the others, RADAR is abbreviated as RA when needed.

6.6.1 Heterogeneity

Because our use case implies that some participants share similar data distributions, we expect RADAR's clustering component to limit the impact of heterogeneity by grouping similar participants together. To evaluate our approach, we compare the partition created by RADAR's clustering algorithm with one where participants are grouped according to their

Table 6.4 – *Effect of different attack configurations (*`100T/U`*) on all baselines.* The Attack Success Rate (ASR) is computed over the targeted classes in targeted attacks, and over all samples otherwise (see **??**). `RA` is `RADAR`, `FG` is `FoolsGold`, `FA` is `FedAvg` (on *all* participants), and `FC` is `FedAvg` ideally clustered per dataset. The ASR of benign runs is provided as a baseline. `RADAR`'s limiting scenario is marked ‡.

Scenario	Mean accuracy (%)				ASR (%)			
	RA	FG	FA_a	FC	RA	FG	FA	FC
Targeted (100T)								
Benign	99.07	55.04	79.49	**99.24**	**0.00**	5.17	5.10	0.09
Lone	99.06	60.51	77.38	**99.22**	**0.00**	93.82	6.73	0.45
Collud. min.	**98.96**	54.64	78.48	98.33	**0.00**	2.97	9.99	53.40
‡ Collud. maj.	**98.28**	85.10	79.40	98.22	73.39	**8.10**	17.65	59.36
Untargeted (100U)								
Benign	99.07	55.04	79.49	**99.24**	0.09	0.39	33.30	**0.06**
Lone	98.96	49.56	78.38	**99.22**	**0.08**	99.89	54.70	0.12
Collud. min.	**98.98**	49.67	72.47	97.69	0.10	**0.04**	44.53	6.26
Collud. maj.	**98.96**	69.09	81.87	75.66	**0.08**	38.98	59.49	94.36

dataset of origin. This partition is presented as Partition A in Table 6.3. The constant Rand Index of 1.0 indicates that all participants are correctly grouped, regardless of the considered evaluation scenario. This validates the idea that similarity between evaluations can be used to regroup participants.

In addition to managing heterogeneity, it is critical that the countermeasures deployed in `RADAR` do not negatively impact performance. Specifically, the reputation system must not unfairly penalize legitimate participants because of their potential differences. Figure 6.4a presents the weights provided by the reputation system for the aggregation. In the `Benign` scenario, the 5 participants originating from the Bot-IoT dataset do have equal weights, confirming that none of them is penalized by the reputation system. Furthermore, Table 6.4 indicates that `RADAR`'s mean accuracy is superior to `FoolsGold`'s and `FedAvg`, as both baselines falter in practical NIID use cases. `RADAR` almost matches the results of `FC`, which is ideally clustered by design. Overall, `FoolsGold`, a reference Byzantine-resilient FL strategy tailored for NIID settings, falters in *practical* NIID settings, where `RADAR` strives.

6.6.2 Handling data quality

Another goal for `RADAR` is to handle contributions of various quality. This objective is mostly represented by scenarios of Category C2 (`Lone`), as we consider that coordinated faults are improbable for legitimate participants. In this configuration, we expect the Byzantine participant to be either, put in a cluster of its own, or penalized by the reputation system. To verify the former, we compare the partition made by `RADAR` with another

where Byzantines are segregated in an additional cluster (see Partition B in Table 6.3). Here, a Rand Index lower than 1.0 implies that Byzantine participants have been grouped with legitimate ones of the same dataset, which is the case in most scenarios of the `Lone` category. However, the noisiest untargeted faults (`Lone >95U`) result in the Byzantine participant being placed in his own separate cluster, thus neutralizing its impact on the other participants. Note that the hyperparameters of the clustering algorithm could be tuned so that attackers with lower *noisiness* would be separated, notably the threshold factor β and the cross-evaluation metric (see Table 6.1).

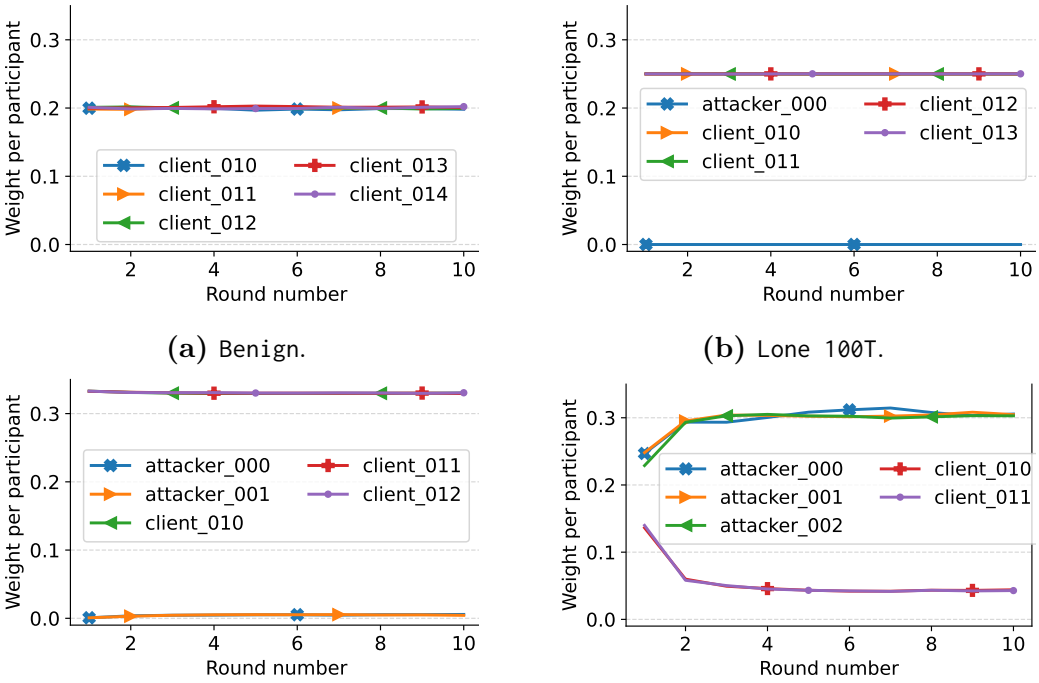
When Byzantine participants are grouped with benign ones, we rely on the reputation system to identify and diminish the impact of their contributions. The weights given by the reputation system can be seen in Figure 6.4b, where the Byzantine client is heavily penalized in the `Lone 100T` scenario. The effect of the clustering and reputation system are also apparent in Table 6.4, where the ASR for both `Lone 100T` and `Lone 100U` are comparable to the benign case, underlining `RADAR` resilience. The results in Figure 6.5 confirm this trend: `RADAR` maintains a low ASR in most configurations. As a result, `RADAR` demonstrates its ability to mitigate isolated Byzantine faults, regardless of their intensity.

The same cannot be said for `FoolsGold`'s, which aims at providing a single global model. Further, by construction, it identifies groups of similar participants as colluding attackers and considers that only the faulty participant is legitimate. This appreciation error leads `FoolsGold` to have the worst attack success rate among all tested baselines, even when compared against the naive `FedAvg` approach.

6.6.3 Label flipping attacks

We evaluate the resistance to label-flipping attacks using two different scenarios. First, we consider that `Colluding Byzantines` can only refer to attackers, as it is unlikely that the very same fault happens over multiple clients at the same time. Second, the `Lone 100U` scenario, as it is similarly unlikely that for an *honest-but-neglectful* participant to misclassify the entirety of its data.

Like discussed in Section 6.6.2, the clustering algorithm separates the noisiest attacks from the rest. This is true regardless of the number of attackers, as confirmed by the results in Table 6.3. For untargeted faults with at least 95% *noisiness*, the Rand Index at round 10 stays equal to 1.0. This means that for those loud attacks, attackers are separated from benign participants, hence negating their poisoning effect. This is a critical result for `RADAR`, as this mitigation occurs for *any number of attackers*, even if they outnumber benign participants. However, the attackers in `Colluding T` scenarios are placed with legitimate participants in the same cluster.



(a) Benign. (b) Lone 100T.

(c) Colluding minority 100T. (d) Colluding majority 100T.

Figure 6.4 – *Aggregation weights ρ_i^r for the participants coming from the BoT-IoT dataset depending on the number of Byzantines (*`100T`*).* Byzantines are correctly penalized when they are a minority, but gain precedence when they become the majority.

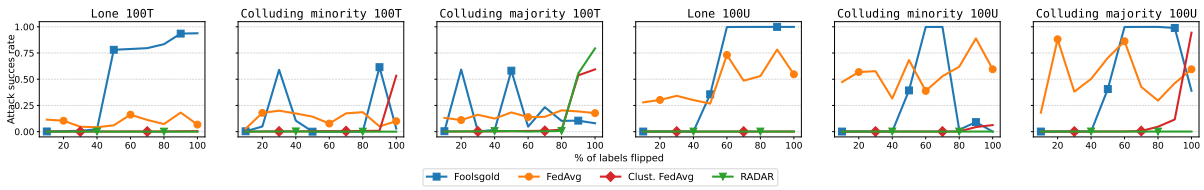


Figure 6.5 – *Attack Success Rate (ASR) of the different baselines.* Even though attackers are a majority, they gain weight precedence only for higher poisoning rates (>90%).

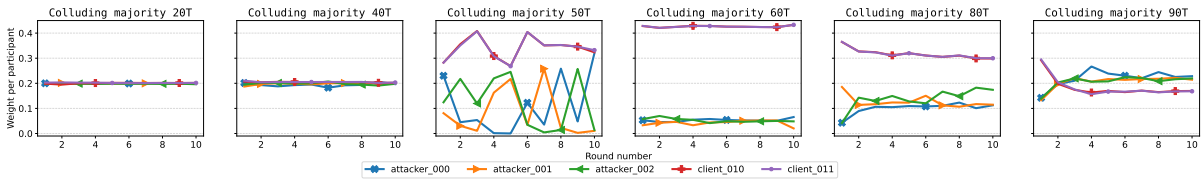


Figure 6.6 – *Aggregation weights ρ_i^r per participant of the poisoned cluster (*`Colluding majority T`*).* Even though attackers are a majority, they gain weight precedence only for higher poisoning rates (≥90%).

Minority of attackers The `Colluding minority` class (Category C3) contains scenarios where 2 out of 5 participants instantiated in Bot-IoT perpetrate label-flipping attacks. Here, the results depicted in Figure 6.4c indicate that the attackers are heavily penalized by the reputation system. This is coherent with the results in Table 6.4 for these scenarios, where we can see that `RADAR` indeed fend off attackers with an ASR of 0.0. Among the other baselines, `FedAvg` is especially affected, since it does not have any protection against such attacks. This is also true for our theoretical baseline `FC`, although the effect is logically limited to participants using the Bot-IoT dataset. `FoolsGold`, on the other hand, detects the attackers since they are similar and thus manages to discard the attack, obtaining a rather low ASR of 2.97%. Unfortunately, it also detects benign members from the other clusters as colluding attackers and thus train on BoT-IoT only, leading to a very low 54.64% accuracy overall.

Majority of attackers The `Colluding majority 100T` scenario, with 3 attackers out of 5 participants, sees the attackers gain precedence. Figure 6.4d clearly illustrates this phenomenon, where the legitimate participants' weights drop as the reputation system favors the attackers. This is a known limit of the reputation system, which favors the majority by construction. This is further illustrated in Figure 6.7: a steeper drop in accuracy and miss rate occurs when attackers outnumber benign participants in one cluster. However, the metric distribution over the participants highlights that the other clusters remain unaffected, and that the majority of benign participants continues to perform well. Furthermore, as illustrated in Figures 6.5 and 6.6, the *noisiness* of attackers must exceed 80% for attackers to poison the cluster's model. Consequently, while this scenario highlights a limitation of `RADAR`, it is significantly constrained.

Impact of the attack timing Additionally, Figure 6.8 depicts how the reputation system reacts to participants that change their *noisiness* over time. Figure 6.8a features a `Colluding minority 100T` scenario where the noisiness drops to 0% at round 3. The system forgives attackers approximately four rounds after they adapted their behavior. This rather short delay depends on the chosen λ history parameter of our reputation system (see Table 6.1). On the contrary, Figure 6.8b showcases `Colluding minority T` attackers going from 0 to 100% noisiness over the course of a few rounds. The reputation system detects and penalizes them at round 5 when the noisiness reaches 60%. This in phase with the conclusions of Figures 6.5 and 6.6: for lower noisiness levels, the attackers have no effect. The reputation system thus detects attackers only when they start to present a threat to the global model's performance.

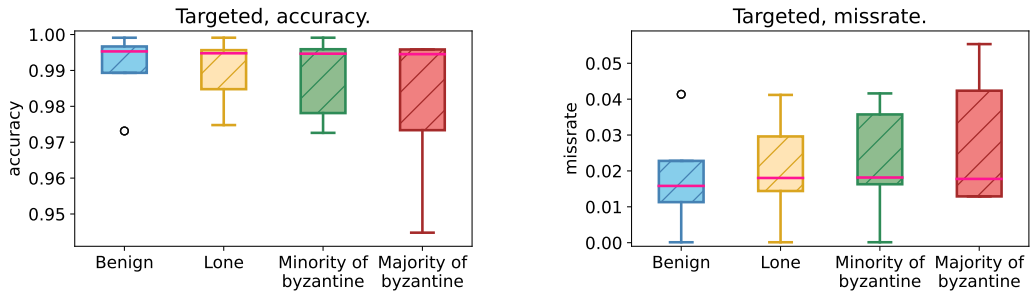


Figure 6.7 – RADAR*'s metric distribution among participants in different scenarios (*100T*).* The accuracy's and miss rate's lower bounds suddenly drop when attackers outnumber benign participants in the affected cluster. Indeed, clients in other clusters are unaffected by the poisoning.

6.6.4 Synthesis

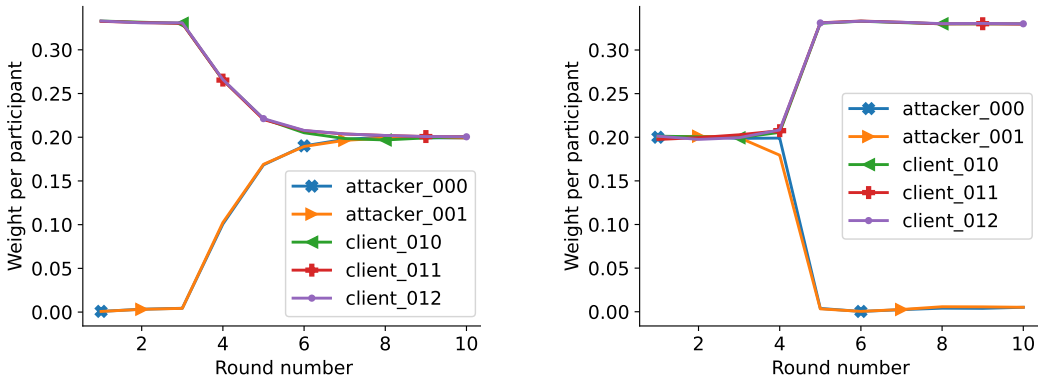
First, the results highlight the relevance of clustering in *practical NIID* use cases, as attacks are confined to the cluster attackers have been assigned to. This is particularly visible in the performance of RADAR and the clustered FedAvg variant, which both maintain high accuracy overall by providing each community with a specific model. This is true even in the presence of Byzantine faults or attackers. However, since FC does not implement any mitigation strategy, its performance quickly degrades with the quality of the contributions, especially in the presence of colluding attackers (as illustrated by Figure 6.5).

The results in Table 6.4 also emphasize on FoolsGold's unsuitability for *practical NIID* use cases, where groups of participants sharing similar distributions can exist. Especially in a Lone scenario, any groups of similar participants are considered as colluding attackers and penalized, leading to high ASR, as only the attacker is considered as legitimate. Similarly, in Colluding majority T/U scenarios, FoolsGold penalizes all the other clusters, leading to a model trained on Bot-IoT only. Overall, RADAR presents the most consistent results, with high accuracy and low ASR in most scenarios, only failing against a majority of extremely *noisy* colluding attackers that still managed to get similar enough to be grouped with benign participants.

6.7 Discussion

The experiments illustrate how RADAR succeeds at identifying attackers in heterogeneous context, thus demonstrating its versatility. In this section, we discuss the limitations and potential consequences of our architecture and propose research directions to close these gaps.

Heterogeneity The experiments conducted in Section 6.6 show that RADAR can handle heterogeneous participants. However, the simulation of the *practical* NIID setting is



(a) Attackers act with 100% *noisiness*, but become benign on round 3.

(b) Attackers start benign, and increase *noisiness* by 20% each round when $r \geq 3$.

Figure 6.8 – *Aggregation weights ρ_i^r per participant of the poisoned cluster (*`Colluding minority T`*).* Attackers are forgiven over time, and the reputation system reacts quickly to newly detected attackers.

limited by the available datasets and the partitioning choices. In RADAR, we use IID partitioning with each of our datasets to create our communities. This choice is motivated by the absence of control over the data distribution of the participants with datasets from different sources. The approach presented in Chapter 7 is expected to cope with this limitation, and thus represent an interesting research direction to further investigate heterogeneous settings.

Generalizability While the experiments are only conducted on intrusion detection datasets, RADAR's design could be used in different use cases regarding the following conditions: (1) parametric local models whose parameters can be aggregated using FL, and (2) local testing sets and relevant metrics allowing participants to evaluate the others models. Since the NIDS use case induces a focus on malicious samples (*i.e.* *positive* values), we choose the F1-score as input for our clustering and reputation algorithms, as it emphasizes on false positives and false negatives. However, RADAR can handle different metrics, for instance the loss of a model during evaluation, particularly relevant for similarity measurements.

Scalability and performance The focus on small-scale collaboration (*i.e.* a few dozens of participants) makes the overhead of the *cross-evaluation* step (Section 6.4.1) practical, and justifies the absence of performance-related metrics in this paper. However, one can question the scalability of the proposed approach in larger scale applications. Indeed, at each round, clients evaluate $|P|$ additional models, which scales linearly with the number of clients. Two new communications are also introduced, one to send the models and one to collect the evaluations. Their size also grows linearly with $|P|$, as the models of all participants must be evaluated. Likewise, we exclude execution-related performance

evaluation such as training time, CPU overhead, or bandwidth consumption. It opens the way to interesting research directions on how to implement and scale RADAR while guarantying its properties.

Evaluation poisoning Attackers could try to poison the evaluations that they provide on other participants to abuse the system. However, the implementation presented in **??** implies that attackers poison both their training and testing sets. Consequently, the evaluations they produce on other participants are directly affected. We thus expect the system to cope with arbitrary poisoning similarly to data poisoning: either by placing the attackers in a different cluster because of their dissimilarity, or by penalizing their reputation.

Information disclosure Because RADAR shares models with the other participants to obtain feedbacks, it can be argued that it revels more information about the participants. This is limited to the participants' models, which are shared without identifiers. However, since clients also receive the global model of their cluster, they can try to estimate the models that belong to their cluster. This remains challenging, as the models are weighted using the reputation score of the participants, which are only available to the server. Comparing the privacy impact of RADAR with those of simpler approaches like FedAvg represents interesting research directions.

6.8 Conclusion

In this chapter, we introduced RADAR, a Federated Learning framework that effectively deals with Byzantine participants, even with heterogeneous data-distributions. To that end, we introduce a cross-evaluation scheme that allows participants to subjectively estimate their pairwise similarities. Based on those measurements, we manage to rebuild the initial participant distribution using hierarchical clustering. Our results confirm that evaluation metrics can indeed be used to assess similarity between participants, without accessing their datasets nor comparing their models statistically.

We further designed a reputation system based on the cross-evaluation results. Our reputation system uses the perceived similarity of participants and their cumulated past results to give a score to each participant inside a cluster. We are able to validate that the combination of the clustering and reputation system can mitigate all tested Byzantines scenarios, with the single exception of targeted attacks where a majority of Byzantines flip more than 80% of their labels. To the best of our knowledge, this is the first reputation system in FL that leverages indirect feedbacks to assess the quality of the participants' contributions.

RADAR is the keystone of this thesis, as it addresses the main challenges of FIDSs in an untrusted and heterogeneous environment. More importantly, it participates in laying out the foundations for the future of FIDSs. Indeed, its intrinsic qualities, notably the indirect feedbacks and personalized model weighting, makes it a suitable candidate for decentralized architectures. In this regard, being able to remove the central server dependency is a key step towards a truly decentralized, trustworthy, and privacy-preserving collaborative machine learning framework. The next chapter will further explore these directions with a discussion on the future of FIDS, and the potential of RADAR in this context.

Chapter 7

Topology Generation for Independent Distributed Datasets

Contents

7.1 Introduction

Over the course of the past chapters, we have seen how the performance of Federated Intrusion Detection Systems (FIDSs) can be impacted by the heterogeneity of the participants' data distributions. Yet, we have been limited in our experiments by the lack of datasets to thoroughly evaluate this aspect of FIDSs. Indeed, the existing public datasets in the literature are typically created using a single network topology, leaving researchers working on distributed approaches with two choices: (i) use the existing datasets and rely on partitioning strategies to simulate the heterogeneity of real-world data distributions; or (ii) apply standardized feature sets on existing public datasets to create training sets coming from independent, siloed infrastructures.

Unfortunately, both of these approaches have limitations. In the first case, the partitioning strategies cannot fully replicate the heterogeneity of real-world data distributions, as data will remain correlated to some extent. Moreover, the partitioning strategies are not always applicable to all datasets, and it requires a deep understanding of the way each dataset is generated to approach realistic data shards. In the second case, the number of clients is limited by the number of public datasets available, narrowing experiments to extremely small-scale federations. Additionally, because all datasets are independent, characterizing the heterogeneity of the data distributions is difficult, leaving little control over the experimental conditions. In Chapter 6, we leveraged a combination of both strategies to simulate *practical* NIID (Non Independent and Identically Distributed) settings, but the results remained limited in realism by the lack of control over the data

distributions.

To close the gap towards more realistic evaluations of FIDSs, we introduce a novel approach to generate heterogeneous network topologies that can be deployed in virtualized environments. Because creating a functional topology from scratch is particularly complex, we propose to compose topologies from a set of predefined building blocks that satisfy at set of user-defined constraints. By leveraging routing protocols and domain name resolution, we can dynamically a large number of topologies that can be used to evaluate the performance of FIDSs an heterogeneous, yet controlled, environment.

The content of this chapter originates from the preliminary work presented at the C&ESAR conference in late 2022 [Léo+22]. Its remainder is structured as follows. We start by laying out the requirements for topology generation in the context of FIDSs in Section 7.2, and review existing works in the field in Section 7.3. We then present our approach to generate topologies in Section 7.4, and evaluate the performance of our tool in Section 7.5. Finally, we discuss the perspectives of our work in **??** before concluding in Section 7.6.

Contributions of this chapter

- A novel approach to generate realistic network topologies for dataset generation by leveraging constraint-based topology composition.
- A benchmark of the number of topologies that can be generated using our approach.
- The foundations for the first truly distributed dataset in intrusion detection, enabling the evaluation of FIDSs in under controlled conditions.

7.2 Requirements

The topic of topology generation has been extensively researched in the late 90s and early 2000s, notably to evaluate the performance of network protocols in large-scale networks [Med+01]. In a structuring survey on network topologies, Haddadi *et al.* [Had+08] synthesized the requirements for network topology generators, from which we extract the following requirements for our use case: representativeness, extensibility, and efficiency.

In addition to these requirements, since the goal of this work is to build topologies for dataset generation, we take inspiration from the literature on Network-based Intrusion Detection System (NIDS) datasets. In their work, Ring, Wunderlich, Scheuring, *et al.* [Rin+19] identify the qualities of a *perfect* dataset. It should: be up-to-date, correctly labeled, and publicly available, contain real network traffic with various attacks and normal

user behavior, and span a long time. Finally, because we strongly believe in the importance of reproducibility in experimental research, we follow the requirements laid out by Uetz *et al.* [Uet+21] for sound experiments. They need to be: valid (*i.e.*, well-defined and unrefutable), controllable (*e.g.*, parameterized), and reproducible (*i.e.*, the same results can be obtained by another group using the author's artefact). Based on these qualities, we derive the following requirements for the topology generator.

— *Representativeness:* the generated topologies must be representative of modern real-world networks and respect their statistical properties.

— *Extensibility:* the tool must allow users to extend its capabilities.

— *Efficiency:* the tools must be efficient to generate large-scale topologies without altering their properties.

— *Validity:* the generated topologies must be exempt of side effects or biases that could alter the results of the experiments.

— *Controllability:* the generator must allow precise control over the differences between the generated topologies.

— *Reproducibility:* the generator must be able to deterministically generate the same topology multiple times.

7.2.1 Controlling Heterogeneity

The major challenge in generating network topologies for FIDSs is to control the heterogeneity of the generated datasets. Notably, the generated topologies and the associated datasets must allow researcher to evaluate the impact of different data-distributions on FIDSs, what identify the aspects of heterogeneity that are the most impactful. Consequently, the generator must allow precise control over the differences between the generated topologies. To this end, we identify five main qualities that characterize the heterogeneity of network topologies: architecture, attack scenarios, hosted services, user behaviors, and maturity.

1. **Architecture.** The network architecture of the topology defines how services are interconnected, how the traffic is captured, and where data collection is performed. For instance, a topology with a single main gateway which captures the traffic of several services on the same network will produce a different dataset when compared with a star-shaped topology with multiple subnets. Appropriated metrics are required to characterize the impact of these differences, *e.g.*, size (number of hosts, of subnets), mean number of hops between a service and the last gateway, and so on.
2. **Attack scenarios.** The literature on intrusion detection is rich with different classes of attacks that generate different patterns of traffic. For instance, a Denial of Service

(DoS) or brute force attack will generate a lot of traffic which will vary depending on the targeted service, *e.g.*, an SSH server, a database over TCP, and so forth.

3. **Hosted services.** Different services can rely on different protocols, and therefore generate different kind of data. For example, a service using TCP will induce connection establishment, and therefore a lot of traffic back-and-forth, whereas something based on UDP will produce a more continuous stream of data. The Internet of Things (IoT) also introduce new kinds of network traffic patterns, with unusual protocols such as CoAP or MQTT. Therefore, different services (and protocols) might have different normal behaviors, causing heterogeneity among participants. The list of considered services must be adapted depending on the considered attack scenarios.
4. **User behaviors.** The behavior of users can also impact the network traffic depending on the considered use case and the type of organizations. For example, a university network will have a lot of students connecting to the internet, whereas a company will have a lot of internal traffic. Other factors like working hours, the sue of a Virtual Private Network (VPN), or the use of a Bring Your Own Device (BYOD) policy can also impact the network traffic.
5. **Maturity.** Security practices vary between organizations, depending on their threat model, previous expertise, and budget. For example, a company might have a dedicated security team, and therefore be able to implement a more mature security policy, whereas a small company might not have the resources to do so. This parameter is important to consider, as it can impact the quality of the dataset, *e.g.*, by having unseen attacks in the training data, supposed to be benign traffic.

User behaviors can be implemented directly in traffic generators and are not considered in the scope of this work. The maturity of the organization can be simulated by two approaches: either by generating different topologies with different security policies and relying on the service description to generate the topologies, or by altering data quality in the generated datasets afterward. Consequently, we focus on the architecture, attack scenarios, and hosted services to define the requirements of the topology generator.

7.3 Related Work

The motivation behind topology generation originates from the need to evaluate network protocols in simulations. In fact, while network topology should not influence the behavior of a protocol, it can significantly impact its performance [Tan+02]. Multiple tools have been developed to generate network topologies at the time, such as `GT-ITM` [CDZ97], `Tiers` [Doa96], or `BRITE` [Med+01]. Tangmunarunkit *et al.* [Tan+02] distinguish two main categories of topology generators: *structural*, which aim at reproducing the structural

properties of the internet and particularly its hierarchical organization, and *degree-based*, which focus on the statistical properties of the network, notably the power-law distribution of the node degrees [FFF99]. Most of these works are more than 20 years old and have been developed to generate topologies for internet-scale networks, which are not directly applicable to the generation of FIDSs datasets.

Recent works on topology generation are rarer and focus on specific use cases. For instance, Laurito *et al.* [Lau+17] developed `TopoGen`, a tool to generate network topologies using Software-Defined Networkings (SDNs). Their approach allows users to programmatically define the network topology using the Ruby programming language, and extract existing topologies from real-world networks. Yet, their approach is limited to SDNs and does not allow automating data generation. Alrumaih and Alenazi [AA23] developed `GENIND`, a tool to generate industrial network topologies. Similarly to us, the authors identified that most existing tools are too focused on internet-inspired and internet-scale topologies, and do not allow generating topologies for specific use cases. Their tool focus on generating topologies for industrial networks, and therefore generates topologies with specific constraints layer-by-layer, before connecting the different sub-topologies together in a multigraph. To the best of our knowledge, no tool has been developed to generate *deployable* network topologies for IT networks, and can be randomized to generate a large number of topologies with common characteristics.

7.4 Topology Generator for Federated IT Networks

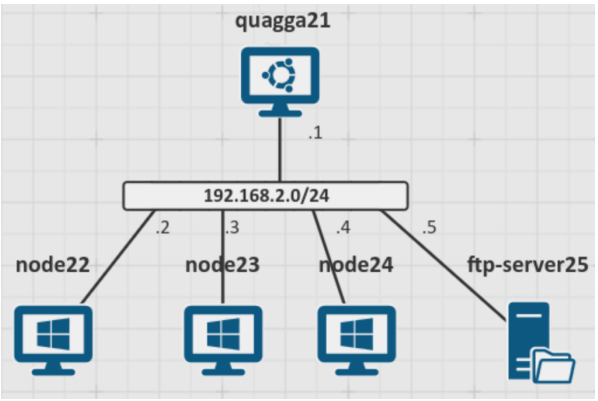
To solve the limitations of existing datasets in the literature, we introduce `FedITN_gen`, a topology generator for federated IT networks. In this section, we present the design and architecture of `FedITN_gen`, before detailing its implementation using Airbus' CyberRange platform [1].

7.4.1 Approach Overview

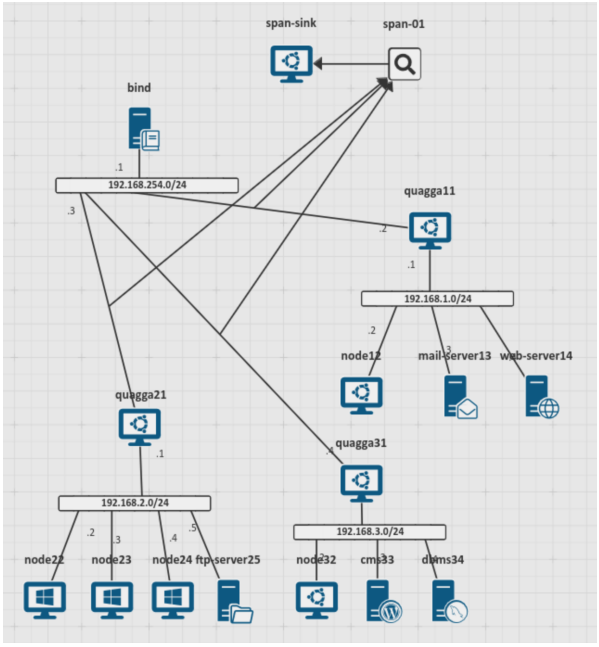
The core idea behind `FedITN_gen` is to compose network topologies by selecting and connecting a set of building blocks that satisfy user-supplied constraints from a library of predefined sub-topologies. While creating this library remains human operated, the composition of the topologies is fully automated, allowing the generation of different topologies with common characteristics. `FedITN_gen` is a two-step algorithm:

1. *Topology selection:* Use constraint programming to find all sets of sub-topologies that satisfy the user-defined constraints, based on a library of predefined sub-topologies.
2. *Topology composition:* For each set, connect the sub-topologies in a tree-like structure starting from the *Master* sub-topology.

1. https://www.cyber.airbus.com/products/cyberrange/



(a) Example of a sub-topology instantiated on Airbus' CyberRange, containing 3 Windows clients and 1 FTP server. The node labeled `quagga21` is the gateway connecting the sub-topology to the rest of the network.



(b) Example of a complete topology composed of 3 sub-topologies. Each sub-topology is connected to the rest of the network through a gateway.

Figure 7.1 – Topology generation with `FedITN_gen`.

7.4.1.1 Topology selection

The topology selection is the most important part of the algorithm, as it requires finding all sets of sub-topologies that satisfy the user-defined constraints. A sub-topology is composed of a subnet, a set of nodes (clients or servers), and a gateway that connects the subnet to the rest of the network. Figure 7.1a shows an example of a sub-topology with 4 nodes. Each sub-topology is formally defined as its `/24` subnet s, which also can serve as its identifier, and a set of clients H_s and services S_s that we generally refer to as nodes, *i.e.* $N_s = H_s \cup S_s$. The gateway g_s is a particular node (not comprised in N_s) that connects the subnet to the rest of the network. A sub-topology can therefore be represented as a tuple $t_s = \langle s, N_s, g_s \rangle$. The topology selection is a Constraint Satisfaction Problem (CSP) where the goal is to find all sets of sub-topologies that satisfy the user-defined constraints. Definition 7.1 defines the concept of a CSP.

While dimensioning the output topologies can easily be translated using numerical boundaries, the constraints on the availability of services and attack scenarios in the sub-topologies are slightly more challenging. We first define the service domain D_{service} as the set of all services available in the library of sub-topologies, and tag each service node with its corresponding name, such as `ftp`, `postfix` (for email), or `ldap`. An attack scenario is defined as another tuple $A_k = \langle \text{srcs} = \{N_1, N_2, \ldots\}, \text{targets} = \{N_{11}, N_{12}, \ldots\} \rangle$ where `srcs` and `targets` are sets of nodes available in the library that are compatible with the

Definition 7.1: Constraint Satisfaction Problem (CSP) [RN21]

A Constraint Satisfaction Problem (CSP) is a tuple $P = \langle X, D, C \rangle$ where:

- $X = \langle X_1, X_2, \ldots, X_n \rangle$ is a set of variables.
- $D = \langle D_1, D_2, \ldots, D_n \rangle$ is a set of non-empty domains, one for each variable.
- $C = \langle C_1, C_2, \ldots, C_n \rangle$ is a set of constraints that specify allowable combinations of values in their domains.

Each constraint $C_j \in C$ is a new tuple $C_j = \langle \chi, R \rangle$ where R is a relation between the variables in $\chi \subseteq X$. Thus, solving a CSP is finding an assignment of values for X in D that satisfies all constraints in C.

attack scenario. For instance, a Man-in-the-Middle (MitM) attack could be represented as $A_{\text{MITM}} = \langle \text{srcs} = \{N_{\text{attaker}}\}, \text{targets} = \{N_1, N_2\} \rangle$. Based on the aforementioned definitions, we can define our Topology Selection Problem (TSP) in Problem 7.1.

Problem 7.1: Topology Selection Problem (TSP)

Given a set of sub-topologies $T = \{t_1, t_2, \ldots, t_n\}$, a set of services $D_{service}$, and a set of attack scenarios $A = \{A_1, A_2, \ldots, A_m\}$, find all sets of sub-topologies $T' \subseteq T$ that satisfy the following:

1. *The number of sub-topologies in T' is between n_{min} and n_{max}.*
2. *The total number of nodes in T' is between h_{min} and h_{max}.*
3. *Each service in $D_{service}$ is available in at least one sub-topology in T'.*
4. *Each attack scenario in A is compatible with at least one sub-topology in T'.*

7.4.1.2 Topology composition

Once the sub-topologies are selected, the next step is to connect them to form a complete IT network. The composition of the topologies is done in a tree-like structure, starting from the *Master* sub-topology. The *Master* sub-topology is a special sub-topology that acts as the root of the tree and contains the necessary services to route traffic between the sub-topologies. Figure 7.1b shows an example of a complete topology composed of 3 sub-topologies. At this point of the algorithm, the sub-topologies are already selected, and the composition is a simple matter of connecting the gateways while respecting the last constraint of tree-depth. Yet, many variations of the same tree can be created, as illustrated in Figure 7.2.

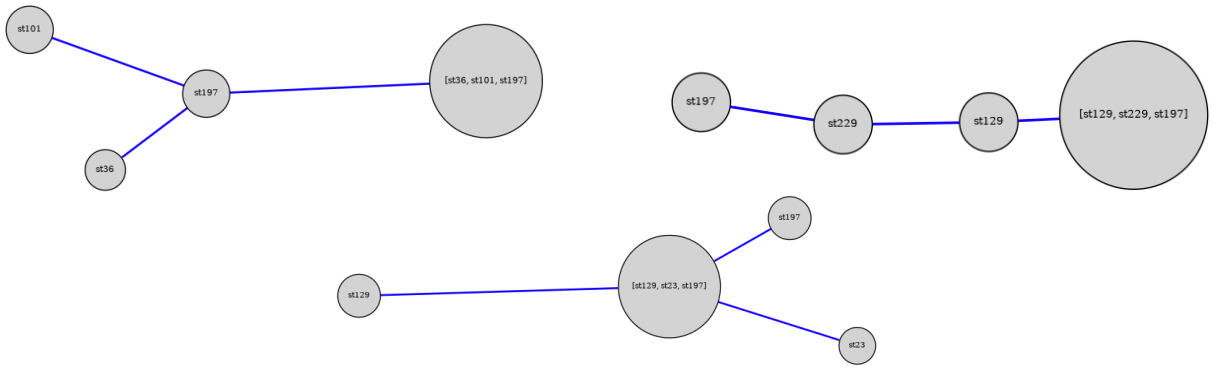


Figure 7.2 – Examples of tree-like topologies composed of 3 sub-topologies. The *Master* sub-topology is represented by the biggest node, and labeled with the IDs of the sub-topologies.

7.4.2 Implementation

In this section, we present parts of the implementation of `FedITN_gen` using Airbus' CyberRange platform. A CyberRange is a virtual environment that simulates a real-world network, allowing users to train and test their cybersecurity skills. Most importantly, such platforms come with a set of templates that can be used to create sub-topologies, and compatible attack scenarios and user traffic generation tools to generate realistic traffic. We implement or topology generator as a Python script that interacts with the CyberRange platform through its API to query the available sub-topologies, services, and attack scenarios. The topology-selection algorithm is implemented using Google's `CP-SAT` solver [2], and we develop a naive recursive algorithm to compose the topologies. The constraints on the availability of services and attack scenarios are implemented as a simple filtering algorithm that removes sub-topologies that do not contain the required services or are not compatible with the attack scenarios. We seed all random operation to ensure deterministic results.

The *Master* sub-topology. The *Master* topology serves as the basis for the composition. It consists of a gateway connecting it to the Internet, a collection point for network logs, a DHCP server for distributing IP addresses, and a DNS server for resolving domain names across all topologies. This is the only topology that will be configured, in particular to allocate the correct domain names to the IPs of the machines hosting the services that need to be accessible. For example: web server (`webserver.local`), mail server (`mailserver.local`), file sharing (`fileserver.local`), etc. This approach makes it possible to define unique domain names for each service, and make them accessible from any topology.

2. https://developers.google.com/optimization/cp/cp_solver

Handling connectivity. Now that all services are available via their domain names, the next step is to ensure that the services are reachable from any sub-topology. To do so, each gateway g_s hosts a DHCP server with a dedicated range to allocate IP addresses child sub-topologies. The DNS configuration of the *Master* topology is propagated to the DHCP server of each gateway, so that the domain names are resolved correctly. To route traffic between the sub-topologies, the gateways also run an OSPF daemon to exchange routing information, announcing their own subnet to the rest of the network. This setup allows to dynamically configure the routing tables of the sub-topologies upon deployment, and ensures that all machines are reachable.

Constraint satisfaction. As noted in Section 7.4.1, the topology selection is a CSP that can be solved using a constraint solver. In particular, we implement our cardinality-related constrains (*i.e.*, the number of sub-topologies and nodes) using the `CP-SAT` solver. This generates all possible combinations of sub-topologies that satisfy the constraints, and we then filter out the ones that do not contain the required services or are not compatible with the attack scenarios. The remaining sets of sub-topologies are then composed into complete topologies using a naive recursive algorithm that walks a tree, starting from the *Master* sub-topology, and randomly assigns children sub-topologies to the gateways. We implement a simple backtracking algorithm to roll back the composition when the tree-depth constraint is not satisfied, and try again from another branch.

7.5 Performance Benchmark

In this section we present some results regarding the performance of `FedITN_gen`, as well as the influence of some of the constraints on performance. To evaluate the tool, we generated 253 sub-topologies with various characteristics which are stored in the library. We then performed a series of experiments to evaluate the performance of `FedITN_gen` in various conditions. We notably measure the impact of the library size, the maximum number of nodes, the tree depth, and the number of service constraints on the performance of `FedITN_gen`. The performance is assessed via the execution time, the number of generated topology sets, and the number of generated final topologies (*i.e.*, tree compositions of the topology sets).

7.5.1 Influence of the library size

We first evaluate the influence of the library size on the performance of `FedITN_gen`. We randomly select a number (in the range 1–29) of sub-topologies from the full library of 253 sub-topologies. For each value of the library size, we perform ten experiments with the parameters fixed as in Table 7.4.

Table 7.1 – Fixed parameters for the library size benchmark.

Parameter	**Value**
Minimum number of nodes	`10`
Maximum number of nodes	`25`
Minimum number of sub-topologies	`2`
Maximum number of sub-topologies	`6`
Services list	`empty`
Attacks list	`empty`
Tree depth	`2`

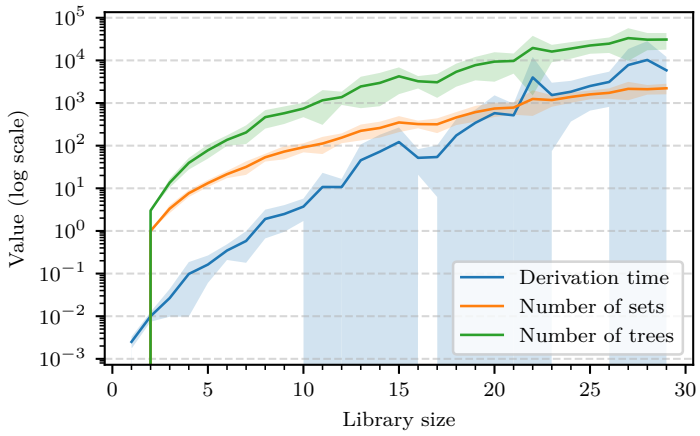


Figure 7.3 – Influence of the library size on the performance of `FedITN_gen`.

Table 7.2 – Fixed parameters for the maximum number of nodes benchmark.

Parameter	Value
Minimum number of nodes	1
Minimum number of sub-topologies	1
Maximum number of sub-topologies	6
Services list	empty
Attacks list	empty
Tree depth	2
Library size	40

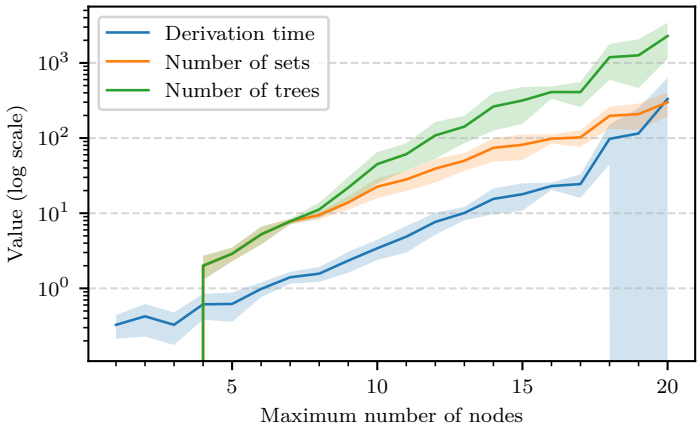


Figure 7.4 – Influence of the maximum number of nodes on the performance of `FedITN_gen`.

Figure 7.3 displays the different metrics on a log scale. We notably observe that the execution time increases exponentially with the library size. This is expected due to the nature of the constraint solver and the tree composition algorithm. The number of sub-topology sets and tree compositions also increase with the library size, but with a lower slope. Note that even with a tree depth of 2, the number of tree compositions is superior to the number of sub-topology sets by a factor of 10.

7.5.2 Influence of the maximum number of nodes

We then evaluate the influence of the maximum number of nodes on the performance of `FedITN_gen`. We vary the maximum number of nodes from 1 to 20, with the other parameters fixed as in Table 7.4. For each value of the maximum number of nodes, we likewise perform ten experiments.

Figure 7.4 displays the different metrics on a log scale. Again, all metrics increase exponentially with the maximum number of nodes. Indeed, this parameter indirectly influences the cardinality of the sub-topology sets generated, and thus the number of tree compositions. Since we kept a tree depth of 2, the number of tree compositions is still

Table 7.3 – Fixed parameters for the tree depth benchmark.

Parameter	Value
Minimum number of nodes	`10`
Maximum number of nodes	`30`
Minimum number of sub-topologies	`2`
Maximum number of sub-topologies	`10`
Services list	`empty`
Attacks list	`empty`
Library size	`20`

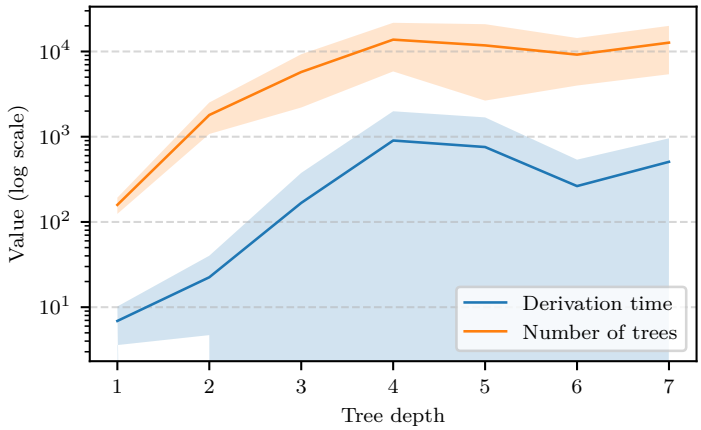


Figure 7.5 – Influence of the tree depth on the performance of `FedITN_gen`.

significantly higher than the number of sub-topology sets. Note that for a number of nodes inferior to 4, there are no solutions, are the topologies in our test library have at least 4 nodes.

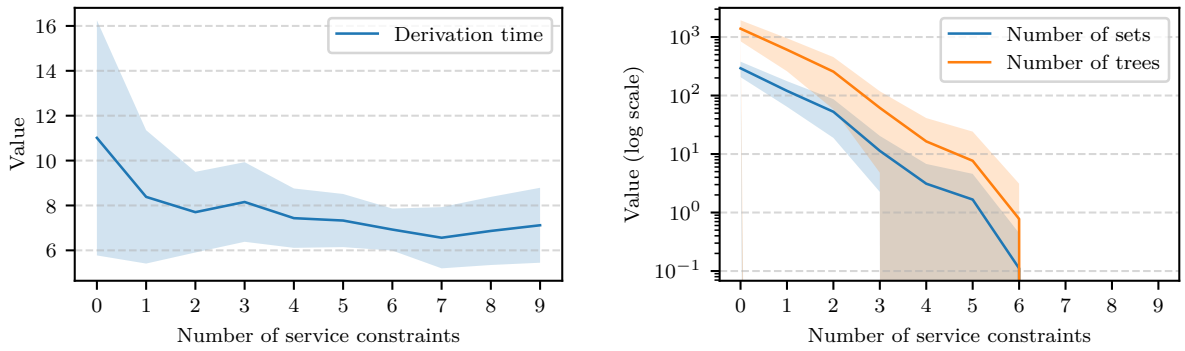
7.5.3 Influence of the tree depth

The next experiment evaluates the influence of the tree depth on `FedITN_gen`'s performance. This parameter has no impact on the number of sub-topology sets, so we only measure the number of tree compositions and the execution time. We vary the tree depth from 1 to 7, with the other parameters fixed as in Table 7.4.

Figure 7.5 displays the execution time and the number of tree compositions on a log scale. Both increase exponentially with the tree depth, as expected, before reaching a plateau with a tree depth of 4. This is due to the fact that the number of tree compositions is limited by the number of sub-topology sets, which in turn depends on the other constraints. Consequently, the variations in the results are only due to the random sampling of the sub-topologies in the library.

Table 7.4 – Fixed parameters for the number of service constraints benchmark.

Parameter	Value
Minimum number of nodes	5
Maximum number of nodes	15
Minimum number of sub-topologies	2
Maximum number of sub-topologies	6
Attacks list	empty
Tree depth	2
Library size	40



(a) Execution time. **(b)** Number of tree compositions.

Figure 7.6 – Influence of the number of service constraints on the performance of `FedITN_gen`.

7.5.4 Influence of the number of service constraints

This last experiment evaluates the influence of the number of service constraints on performance. The library of sub-topologies is generated with a fixed number of available services, namely: `ldap`, `dbms`, `cms`, `dns`, `mail`, `syslog_server`, `web`, `ftp`, `proxy`, and `cloud_storage`. We vary the number of services constraints from 1 to 9, with the other parameters fixed as in Table 7.4.

Figures 7.6 and 7.6b display the execution time and the numbers of sets and tree generations, respectively. Unlike the other experiments, the execution time in almost constant here, due to the way these constraints are handled. While the other constraints are implemented as exploration problems, the service constraints are applied by pruning incompatible topology sets. This is why the number of sets and trees are progressively decreasing as the number of service constraints increases. By the 6^{th} constraint, the problem becomes infeasible, as there are no sub-topologies that satisfy all the constraints. Increasing the maximum number of nodes and sub-topologies would allow for more solutions, but would also increase the execution time.

7.6 Conclusion and Takeaways

In this chapter, we presented `FedITN_gen`, a tool to generate heterogeneous network topologies for generating FIDSs datasets. Because generating such topologies from scratch is impractical, we propose to build a library of predefined sub-topologies that can be combined to form larger topologies according to a set of constraints. We implement a first prototype of `FedITN_gen` to validate the feasibility of the approach and to evaluate its performance. Due to the combinatorial nature of the problem, we rely on a constraint solver to find valid combinations of sub-topologies. Yet, the derivation time of our tool currently scales exponentially with most of the parameters. Meanwhile, the number of generated topologies is also exponential, making `FedITN_gen` a powerful tool to generate a large number of topologies while controlling their heterogeneity.

Perspectives The current implementation of `FedITN_gen` is a first step towards a more complete tool that would support data generation. We believe that having independently generated datasets that are comparable with the state of the art, while allowing finer controls over the heterogeneity of the data, would enable addressing some of the current open challenges in the literature:

(a) *performance against heterogeneity*: the ability for the federation to maintain high performance with heterogeneous participants;

(b) *knowledge transfer between clients*: the ability for one client to recognize patterns that are absent from its local training data;

(c) *model adaptability*: the ability of a local model to evolve in time, and to adapt to new devices in the local network;

(d) *generation capability*: the ability for a local model to correctly characterize behavior for similar but different services.

Future Work `FedITN_gen` is currently a preliminary prototype that we plan to improve in several ways. The first and obvious improvement is to pursue the implementation of the tool to support the automated execution of scenarios (both attacks and legitimate traffic generation) to generate datasets. Another lead for improvement lies in the optimization of the constraint solver, which is currently the bottleneck of the tool. This is critical to introduce finer constraints and allow for more complex topologies. Finally, while we identified as a requirement the respect of the statistical properties of real-world IT networks, this currently remains an open challenge. Consequently, future works include the study and identification of said properties in real-world deployments, and the integration of these properties in the generation process.

Chapter 8

Conclusion

BIBLIOGRAPHY

[16] *Directive (EU) 2016/1148 of 6 July 2016 Concerning Measures for a High Common Level of Security of Network and Information Systems across the Union*, 2016, URL: https://eur-lex.europa.eu/eli/dir/2016/1148/oj.

[22] *Directive (EU) 2022/2555 of the European Parliament and of the Council of 14 December 2022 on Measures for a High Common Level of Cybersecurity across the Union, Amending Regulation (EU) No 910/2014 and Directive (EU) 2018/1972, and Repealing Directive (EU) 2016/1148 (NIS 2 Directive)*, Dec. 14, 2022, URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32022L2555 (visited on 05/26/2024).

[AA23] Thuraya N. I. Alrumaih and Mohammed J. F. Alenazi, « GENIND: An Industrial Network Topology Generator », *in*: *Alexandria Engineering Journal* 79 (Sept. 15, 2023), pp. 56–71, ISSN: 1110-0168, DOI: 10.1016/j.aej.2023.07.062, URL: https://www.sciencedirect.com/science/article/pii/S111001682300649X (visited on 07/08/2024).

[AAA16] Iman Almomani, Bassam Al-Kasasbeh, and Mousa AL-Akhras, « WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks », *in*: *Journal of Sensors* 2016 (2016), pp. 1–16, ISSN: 1687-725X, 1687-7268, DOI: 10.1155/2016/4731953, URL: https://www.hindawi.com/journals/js/2016/4731953/ (visited on 10/25/2021).

[ACA20] Noor Ali Al-Athba Al-Marri, Bekir S. Ciftler, and Mohamed M. Abdallah, « Federated Mimic Learning for Privacy Preserving Intrusion Detection », *in*: *2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), May 2020, pp. 1–6, DOI: 10.1109/BlackSeaCom48709.2020.9234959, URL: https://ieeexplore.ieee.org/document/9234959 (visited on 04/12/2024).

[ACM20] ACM, *Artifact Review and Badging v1.1*, Aug. 24, 2020, URL: https://www.acm.org/publications/policies/artifact-review-and-badging-current (visited on 08/17/2022).

[Agr+22] Shaashwat Agrawal *et al.*, « Federated Learning for Intrusion Detection System: Concepts, Challenges and Future Directions », *in*: *Computer Communications* 195 (Nov. 1, 2022), pp. 346–361, ISSN: 0140-3664, DOI: 10.1016/j.comcom.2022.09.012, URL: https://www.sciencedirect.com/science/article/pii/S0140366422003516 (visited on 06/09/2024).

[Ala+21] Mamoun Alazab *et al.*, « Federated Learning for Cybersecurity: Concepts, Challenges and Future Directions », *in*: *IEEE Transactions on Industrial Informatics* (2021), pp. 1–1, ISSN: 1551-3203, 1941-0050, DOI: 10/gnm4dj, URL: https://ieeexplore.ieee.org/document/9566732/ (visited on 12/02/2021).

[Ale+20] Mohammed Aledhari *et al.*, « Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications », *in*: *IEEE Access* 8 (2020), pp. 140699–140725, ISSN: 2169-3536, DOI: 10.1109/ACCESS.2020.3013541, URL: https://ieeexplore.ieee.org/document/9153560/.

[Ali+18] Janne Ali-Tolppa *et al.*, « SELF-HEALING AND RESILIENCE IN FUTURE 5G COGNITIVE AUTONOMOUS NETWORKS », *in*: *2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K)*, 2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K), Santa Fe: IEEE, Nov. 2018, pp. 1–8, ISBN: 978-92-61-26921-0, DOI: 10.23919/ITU-WT.2018.8598115, URL: https://ieeexplore.ieee.org/document/8598115/ (visited on 03/31/2022).

[ALL21] Sana Awan, Bo Luo, and Fengjun Li, « CONTRA: Defending Against Poisoning Attacks in Federated Learning », *in*: *Computer Security – ESORICS 2021*, ed. by Elisa Bertino, Haya Shulman, and Michael Waidner, Lecture Notes in Computer Science, Cham: Springer International Publishing, 2021, pp. 455–475, ISBN: 978-3-030-88418-5, DOI: 10.1007/978-3-030-88418-5_22.

[And+18] Elli Androulaki *et al.*, « Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains », *in*: *Proceedings of the Thirteenth EuroSys Conference*, New York, NY, USA: ACM, Apr. 23, 2018, pp. 1–15, ISBN: 978-1-4503-5584-1, DOI: 10.1145/3190508.3190538, URL: https://dl.acm.org/doi/10.1145/3190508.3190538.

[Aou+22] Ons Aouedi, Kandaraj Piamrat, Guillaume Muller, *et al.*, « Intrusion Detection for Softwarized Networks with Semi-supervised Federated Learning », *in*: *ICC 2022 - IEEE International Conference on Communications*, ICC 2022 - IEEE International Conference on Communications, May 2022, pp. 5244–5249, DOI: 10.1109/ICC45855.2022.9839042, URL: https://ieeexplore.ieee.org/document/9839042 (visited on 04/25/2024).

[APB20] Ons Aouedi, Kandaraj Piamrat, and Dhruvjyoti Bagadthey, « A Semi-supervised Stacked Autoencoder Approach for Network Traffic Classification », *in*: *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, 2020 IEEE 28th International Conference on Network Protocols (ICNP), Oct. 2020, pp. 1–6, DOI: 10.1109/ICNP49622.2020.9259390, URL: https://ieeexplore.ieee.org/document/9259390 (visited on 06/19/2024).

[Arp+22] Daniel Arp *et al.*, « Dos and Don'ts of Machine Learning in Computer Security », *in*: *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA: USENIX Association, Aug. 2022, pp. 3971–3988, ISBN: 978-1-939133-31-1, URL: https://www.usenix.org/conference/usenixsecurity22/presentation/arp.

[Baj+17] Vaibhav Bajpai *et al.*, « Challenges with Reproducibility », *in*: *Proceedings of the Reproducibility Workshop*, SIGCOMM '17: ACM SIGCOMM 2017 Conference, Los Angeles CA USA: ACM, Aug. 11, 2017, pp. 1–4, ISBN: 978-1-4503-5060-0, DOI: 10.1145/3097766.3097767, URL: https://dl.acm.org/doi/10.1145/3097766.3097767 (visited on 08/12/2022).

[Ber+18] Jeremy Bernstein *et al.*, « signSGD: Compressed Optimisation for Non-Convex Problems », *in*: *Proceedings of the 35th International Conference on Machine Learning*, ed. by Jennifer Dy and Andreas Krause, vol. 80, Proceedings of Machine Learning Research, PMLR, July 10–15, 2018, pp. 560–569, URL: https://proceedings.mlr.press/v80/bernstein18a.html.

[Beu+20] Daniel J Beutel *et al.*, « Flower: A Friendly Federated Learning Research Framework », 2020, arXiv: 2007.14390.

[BFA20] Christopher Briggs, Zhong Fan, and Peter Andras, « Federated Learning with Hierarchical Clustering of Local Updates to Improve Training on Non-IID Data », *in*: *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020 International Joint Conference on Neural Networks (IJCNN), July 2020, pp. 1–9, DOI: 10.1109/IJCNN48605.2020.9207469.

[BG16] Anna L. Buczak and Erhan Guven, « A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection », *in*: *IEEE Communications Surveys Tutorials* 18.*2* (2016), pp. 1153–1176, ISSN: 1553-877X, DOI: 10.1109/COMST.2015.2494502.

[BG17] Suman Sankar Bhunia and Mohan Gurusamy, « Dynamic Attack Detection and Mitigation in IoT Using SDN », *in*: *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), Melbourne, VIC: IEEE, Nov. 2017, pp. 1–6, ISBN: 978-1-5090-6796-1, DOI: 10.1109/ATNAC.2017.8215418, URL: http://ieeexplore.ieee.org/document/8215418/ (visited on 04/09/2022).

[Bha+19] Arjun Nitin Bhagoji *et al.*, « Analyzing Federated Learning through an Adversarial Lens », *in*: *Proceedings of the 36th International Conference on Machine Learning*, International Conference on Machine Learning, PMLR, May 24, 2019, pp. 634–643, URL: https://proceedings.mlr.press/v97/bhagoji19a.html (visited on 02/23/2023).

[Big+14] Elaheh Biglar Beigi *et al.*, « Towards Effective Feature Selection in Machine Learning-Based Botnet Detection Approaches », *in*: *2014 IEEE Conference on Communications and Network Security*, 2014 IEEE Conference on Communications and Network Security (CNS), San Francisco, CA, USA: IEEE, Oct. 2014, pp. 247–255, ISBN: 978-1-4799-5890-0, DOI: 10.1109/CNS.2014.6997492, URL: https://ieeexplore.ieee.org/document/6997492 (visited on 10/25/2021).

[Bla+17] Peva Blanchard *et al.*, « Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent », *in*: *Advances in Neural Information Processing Systems* 30 (2017).

[Bon+17] Keith Bonawitz, Vladimir Ivanov, *et al.*, « Practical Secure Aggregation for Privacy-Preserving Machine Learning », *in*: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA: ACM, Oct. 30, 2017, pp. 1175–1191, ISBN: 978-1-4503-4946-8, DOI: 10.1145/3133956.3133982, URL: https://dl.acm.org/doi/10.1145/3133956.3133982.

[Bon+19] Keith Bonawitz, Hubert Eichner, *et al.*, « Towards Federated Learning at Scale: System Design », *in*: *arXiv* (Feb. 4, 2019), URL: http://arxiv.org/abs/1902.01046.

[Cai+22] Luxin Cai *et al.*, « Cluster-Based Federated Learning Framework for Intrusion Detection », *in*: *2022 IEEE 13th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, 2022 IEEE 13th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), Nov. 2022, pp. 1–6, DOI: 10.1109/PAAP56126.2022.10010553, URL: https://ieeexplore.ieee.org/abstract/document/10010553 (visited on 04/12/2024).

[Cam+22] Enrique Mármol Campos *et al.*, « Evaluating Federated Learning for Intrusion Detection in Internet of Things: Review and Challenges », *in*: *Computer Networks* 203 (Feb. 11, 2022), p. 108661, ISSN: 1389-1286, DOI: 10.1016/j.comnet.2021.108661, URL: https://www.sciencedirect.com/science/article/pii/S1389128621005405 (visited on 04/25/2024).

[Cao+22] Xiaoyu Cao *et al.*, *FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping*, Apr. 11, 2022, DOI: 10.48550/arXiv.2012.13995, arXiv: 2012.13995 [cs], URL: http://arxiv.org/abs/2012.13995 (visited on 08/09/2022), pre-published.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar, « Anomaly Detection: A Survey », *in*: *ACM Computing Surveys* 41.*3* (July 2009), pp. 1–58, ISSN: 0360-0300, 1557-7341, DOI: 10.1145/1541880.1541882, URL: https://dl.acm.org/doi/10.1145/1541880.1541882 (visited on 03/20/2022).

[CDZ97] K.L. Calvert, M.B. Doar, and E.W. Zegura, « Modeling Internet Topology », *in*: *IEEE Communications Magazine* 35.*6* (June 1997), pp. 160–163, ISSN: 1558-1896, DOI: 10.1109/35.587723, URL: https://ieeexplore.ieee.org/document/587723 (visited on 07/07/2024).

[Cet+19] Burak Cetin *et al.*, « Federated Wireless Network Intrusion Detection », *in*: *2019 IEEE International Conference on Big Data (Big Data)*, 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA: IEEE, Dec. 2019, pp. 6004–6006, ISBN: 978-1-72810-858-2, DOI: 10.1109/BigData47090.2019.9005507, URL: https://ieeexplore.ieee.org/document/9005507/ (visited on 10/25/2021).

[Cha+19] Nadia Chaabouni *et al.*, « Network Intrusion Detection for IoT Security Based on Learning Techniques », *in*: *IEEE Communications Surveys & Tutorials* 21.*3* (2019), pp. 2671–2701, ISSN: 1553-877X, DOI: 10.1109/COMST.2019.2896380, URL: https://ieeexplore.ieee.org/document/8629941/.

[Che+11] Dong Chen, Guiran Chang, *et al.*, « TRM-IoT: A Trust Management Model Based on Fuzzy Reputation for Internet of Things », *in*: *Computer Science and Information Systems* 8.*4* (2011), pp. 1207–1228, ISSN: 1820-0214, 2406-1018, DOI: 10.2298/CSIS110303056C, URL: https://doiserbia.nb.rs/Article.aspx?ID=1820-02141100056C (visited on 07/03/2024).

[Che+20] Zhuo Chen, Na Lv, *et al.*, « Intrusion Detection for Wireless Edge Networks Based on Federated Learning », *in*: *IEEE Access* 8 (2020), pp. 217463–217472, ISSN: 2169-3536, DOI: 10.1109/ACCESS.2020.3041793, URL: https://ieeexplore.ieee.org/document/9274294/ (visited on 10/25/2021).

[Che+21] Zheyi Chen, Pu Tian, *et al.*, « Zero Knowledge Clustering Based Adversarial Mitigation in Heterogeneous Federated Learning », *in*: *IEEE Transactions on Network Science and Engineering* 8.*2* (Apr. 2021), pp. 1070–1083, ISSN: 2327-4697, DOI: 10.1109/TNSE.2020.3002796.

[Che+22] Yanyu Cheng *et al.*, « Federated Transfer Learning With Client Selection for Intrusion Detection in Mobile Edge Computing », *in*: *IEEE Communications Letters* 26.*3* (Mar. 2022), pp. 552–556, ISSN: 1558-2558, DOI: 10.1109/LCOMM.2022.3140273, URL: https://ieeexplore.ieee.org/abstract/document/9668958 (visited on 04/12/2024).

[CJ20] Davide Chicco and Giuseppe Jurman, « The Advantages of the Matthews Correlation Coefficient (MCC) over F1 Score and Accuracy in Binary Classification Evaluation », *in*: *BMC Genomics* 21.*1* (Dec. 2020), p. 6, ISSN: 1471-2164, DOI: 10.1186/s12864-019-6413-7, URL: https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-019-6413-7 (visited on 03/11/2022).

[CK21] Zachary Charles and Jakub Konecny, « Convergence and Accuracy Trade-Offs in Federated Learning and Meta-Learning », *in*: (2021), p. 11.

[Cla04] Benoît Claise, *Cisco Systems NetFlow Services Export Version 9*, RFC 3954, RFC Editor, Oct. 2004, DOI: 10.17487/RFC3954, URL: https://www.rfc-editor.org/info/rfc3954.

[Cun+24] Helio N. Cunha Neto *et al.*, « FedSBS: Federated-Learning Participant-Selection Method for Intrusion Detection Systems », *in*: *Computer Networks* 244 (May 1, 2024), p. 110351, ISSN: 1389-1286, DOI: 10.1016/j.comnet.2024.110351, URL: https://www.sciencedirect.com/science/article/pii/S138912862400183X (visited on 04/12/2024).

[CZY20] Yang Chen, Junzhe Zhang, and Chai Kiat Yeo, « Network Anomaly Detection Using Federated Deep Autoencoding Gaussian Mixture Model », *in*: *Machine Learning for Networking*, ed. by Selma Boumerdassi, Éric Renault, and Paul Mühlethaler, Cham: Springer International Publishing, 2020, pp. 1–14, ISBN: 978-3-030-45778-5.

[DAF18] Rohan Doshi, Noah Apthorpe, and Nick Feamster, « Machine Learning DDoS Detection for Consumer Internet of Things Devices », *in*: *2018 IEEE Security and Privacy Workshops (SPW)* (Ml Apr. 11, 2018), pp. 29–35, DOI: 10.1109/SPW.2018.00013, URL: https://ieeexplore.ieee.org/document/8424629/.

[dCal+23] Francisco Lopes de Caldas Filho *et al.*, « Botnet Detection and Mitigation Model for IoT Networks Using Federated Learning », *in*: *Sensors* 23.*14* (14 Jan. 2023), p. 6305, ISSN: 1424-8220, DOI: 10.3390/s23146305, URL: https://www.mdpi.com/1424-8220/23/14/6305 (visited on 04/12/2024).

[dCar+23] Gustavo de Carvalho Bertoli *et al.*, « Generalizing Intrusion Detection for Heterogeneous Networks: A Stacked-Unsupervised Federated Learning Approach », *in*: *Computers & Security* 127 (Apr. 1, 2023), p. 103106, ISSN: 0167-4048, DOI: 10.1016/j.cose.2023.103106, URL: https://www.sciencedirect.com/science/article/pii/S0167404823000160 (visited on 03/14/2023).

[dCos+19] Kelton A.P. da Costa *et al.*, « Internet of Things: A Survey on Machine Learning-Based Intrusion Detection Approaches », *in*: *Computer Networks* 151 (Mar. 2019), pp. 147–157, ISSN: 13891286, DOI: 10.1016/j.comnet.2019.01.023, URL: https://doi.org/10.1016/j.comnet.2019.01.023.

[Den+21] Yongheng Deng, Feng Lyu, Ju Ren, Yi-Chao Chen, *et al.*, « FAIR: Quality-Aware Federated Learning with Precise User Incentive and Model Aggregation », *in*: *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, Vancouver, BC, Canada: IEEE, May 10, 2021, pp. 1–10, ISBN: 978-1-66540-325-2, DOI: 10.1109/INFOCOM42981.2021.9488743, URL: https://ieeexplore.ieee.org/document/9488743/ (visited on 03/27/2024).

[Den+22] Yongheng Deng, Feng Lyu, Ju Ren, Huaqing Wu, *et al.*, « AUCTION: Automated and Quality-Aware Client Selection Framework for Efficient Federated Learning », *in*: *IEEE Transactions on Parallel and Distributed Systems* 33.*8* (Aug. 2022), pp. 1996–2009, ISSN: 1558-2183, DOI: 10.1109/TPDS.2021.3134647, URL: https://ieeexplore.ieee.org/abstract/document/9647925 (visited on 03/27/2024).

[Doa96] M.B. Doar, « A Better Model for Generating Test Networks », *in*: *Proceedings of GLOBECOM'96. 1996 IEEE Global Telecommunications Conference*, Proceedings of GLOBECOM'96. 1996 IEEE Global Telecommunications Conference, vol. MiniConfInternet, Nov. 1996, pp. 86–93, DOI: 10.1109/GLOCOM.1996.586131, URL: https://ieeexplore.ieee.org/abstract/document/586131 (visited on 07/07/2024).

[Dol06] Eelco Dolstra, « The Purely Functional Software Deployment Model », S.l.: s.n., 2006.

[Don+20] Ye Dong *et al.*, « EaSTFLy: Efficient and Secure Ternary Federated Learning », *in*: *Computers & Security* 94 (July 2020), p. 101824, ISSN: 01674048, DOI: 10.1016/j.cose.2020.101824, URL: https://linkinghub.elsevier.com/retrieve/pii/S0167404820300985 (visited on 05/18/2021).

[Dou02] John R. Douceur, « The Sybil Attack », *in*: *Peer-to-Peer Systems*, ed. by Peter Druschel, Frans Kaashoek, and Antony Rowstron, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2002, pp. 251–260, ISBN: 978-3-540-45748-0, DOI: 10.1007/3-540-45748-8_24.

[Dra+16] Gerard Draper-Gil *et al.*, « Characterization of Encrypted and VPN Traffic Using Time-related Features: » *in*: *Proceedings of the 2nd International Conference on Information Systems Security and Privacy*, 2nd International Conference on Information Systems Security and Privacy, Rome, Italy: SCITEPRESS - Science and and Technology Publications, 2016, pp. 407–414, ISBN: 978-989-758-167-0, DOI: 10.5220/0005740704070414, URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005740704070414 (visited on 10/15/2021).

[Duy+21] Phan The Duy *et al.*, « Federated Learning-Based Intrusion Detection in SDN-enabled IIoT Networks », *in*: *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, 2021 8th NAFOSTED Conference on Information and Computer Science (NICS), Dec. 2021, pp. 424–429, DOI: 10.1109/NICS54270.2021.9701525, URL: https://ieeexplore.ieee.org/abstract/document/9701525 (visited on 04/12/2024).

[EA10] Muna Elsadig and Azween Abduallah, « Biological Intrusion Prevention and Self-Healing Model for Network Security », *in*: *2010 Second International Conference on Future Networks*, 2010 Second International Conference on Future Networks (ICFN 2010), Sanya, Hainan: IEEE, Jan. 2010, pp. 337–342, ISBN: 978-1-4244-5666-6 978-0-7695-3940-9, DOI: 10.1109/ICFN.2010.103, URL: https://ieeexplore.ieee.org/document/5431824/ (visited on 03/31/2022).

[ENI14] ENISA, *Actionable Information for Security Incident Response*, 2014, pp. 1–79.

[EO11] Huwaida Tagelsir Elshoush and Izzeldin Mohamed Osman, « Alert Correlation in Collaborative Intelligent Intrusion Detection Systems—A Survey », *in*: *Applied Soft Computing*, Soft Computing for Information System Security 11.7 (Oct. 1, 2011), pp. 4349–4365, ISSN: 1568-4946, DOI: 10.1016/j.asoc.2010.12.004, URL: https://www.sciencedirect.com/science/article/pii/S156849461000311X (visited on 06/22/2024).

[ERJ21] Gints Engelen, Vera Rimmer, and Wouter Joosen, « Troubleshooting an Intrusion Detection Dataset: The CICIDS2017 Case Study », *in*: *2021 IEEE Security and Privacy Workshops (SPW)*, 2021 IEEE Security and Privacy Workshops (SPW), May 2021, pp. 7–12, DOI: 10.1109/SPW53761.2021.00009, URL: https://ieeexplore.ieee.org/document/9474286 (visited on 06/14/2024).

[Eus18] Sébastien Eustace, *Python Dependency Management and Packaging Made Easy*, Poetry, 2018, URL: https://python-poetry.org/ (visited on 07/03/2024).

[Fan+20a] Yulin Fan *et al.*, « IoTDefender: A Federated Transfer Learning Intrusion Detection Framework for 5G IoT », *in*: *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*, 2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE), Guangzhou, China: IEEE, Dec. 2020, pp. 88–95, ISBN: 978-1-66540-396-2, DOI: 10.1109/BigDataSE50710.2020.00020, URL: https://ieeexplore.ieee.org/document/9343358/ (visited on 10/04/2021).

[Fan+20b] Minghong Fang *et al.*, « Local Model Poisoning Attacks to Byzantine-Robust Federated Learning », *in*: 29th USENIX Security Symposium (USENIX Security 20), 2020, pp. 1605–1622, ISBN: 978-1-939133-17-5, URL: https://www.usenix.org/conference/usenixsecurity20/presentation/fang (visited on 02/23/2023).

[Far+20] Omair Faraj *et al.*, « Taxonomy and Challenges in Machine Learning-Based Approaches to Detect Attacks in the Internet of Things », *in*: *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES 2020: The 15th International Conference on Availability, Reliability and Security, Virtual Event Ireland: ACM, Aug. 25, 2020, pp. 1–10, ISBN: 978-1-4503-8833-7, DOI: 10.1145/3407023.3407048, URL: https://dl.acm.org/doi/10.1145/3407023.3407048 (visited on 06/23/2021).

[Fer+22] Mohamed Amine Ferrag *et al.*, « Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning », *in*: *IEEE Access* 10 (2022), pp. 40281–40306, ISSN: 2169-3536, DOI: 10.1109/ACCESS.2022.3165809, URL: https://ieeexplore.ieee.org/document/9751703 (visited on 04/12/2024).

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos, « On Power-Law Relationships of the Internet Topology », *in*: *SIGCOMM Comput. Commun. Rev.* 29.*4* (Aug. 30, 1999), pp. 251–262, ISSN: 0146-4833, DOI: 10.1145/316194.316229, URL: https://doi.org/10.1145/316194.316229 (visited on 07/07/2024).

[Flo24] Flower Labs GmbH., `fedavg.py` *(Flower)*, https://github.com/adap/flower/blob/main/src/py/flwr/server/strategy/fedavg.py, Jan. 5, 2024.

[FNS22] Elena Fedorchenko, Evgenia Novikova, and Anton Shulepov, « Comparative Review of the Intrusion Detection Systems Based on Federated Learning: Advantages and Open Challenges », *in*: *Algorithms* 15.*7* (7 July 2022), p. 247, ISSN: 1999-4893, DOI: 10.3390/a15070247, URL: https://www.mdpi.com/1999-4893/15/7/247 (visited on 04/24/2024).

[FPS10] Gianluigi Folino, Clara Pizzuti, and Giandomenico Spezzano, « An Ensemble-Based Evolutionary Framework for Coping with Distributed Intrusion Detection », *in*: *Genetic Programming and Evolvable Machines* 11.*2* (June 1, 2010), pp. 131–

146, ISSN: 1573-7632, DOI: 10.1007/s10710-010-9101-6, URL: https://doi.org/10.1007/s10710-010-9101-6 (visited on 06/23/2024).

[Fri+23] Othmane Friha *et al.*, « 2DF-IDS: Decentralized and Differentially Private Federated Learning-Based Intrusion Detection System for Industrial IoT », *in*: *Computers & Security* 127 (Apr. 1, 2023), p. 103097, ISSN: 0167-4048, DOI: 10.1016/j.cose.2023.103097, URL: https://www.sciencedirect.com/science/article/pii/S016740482300007X (visited on 04/12/2024).

[FS16] Gianluigi Folino and Pietro Sabatino, « Ensemble Based Collaborative and Distributed Intrusion Detection Systems: A Survey », *in*: *Journal of Network and Computer Applications* 66 (May 2016), pp. 1–16, ISSN: 10848045, DOI: 10.1016/j.jnca.2016.03.011, URL: https://linkinghub.elsevier.com/retrieve/pii/S1084804516300248 (visited on 06/22/2024).

[Fun+11] Carol J Fung, Jie Zhang, *et al.*, « Dirichlet-Based Trust Management for Effective Collaborative Intrusion Detection Networks », *in*: *IEEE Transactions on Network and Service Management* 8.*2* (June 2011), pp. 79–91, ISSN: 1932-4537, DOI: 10.1109/TNSM.2011.050311.100028.

[FYB19] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh, `deep-fg/` *(FoolsGold)*, https://github.com/DistributedML/FoolsGold/tree/master/deep-fg, May 15, 2019.

[FYB20] Clement Fung, Chris J.M. M Yoon, and Ivan Beschastnikh, « The Limitations of Federated Learning in Sybil Settings », *in*: *23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020)*, San Sebastian: {USENIX} Association, Oct. 2020, pp. 301–316, ISBN: 978-1-939133-18-2, URL: https://www.usenix.org/conference/raid2020/presentation/fung.

[Gar+09] P. García-Teodoro *et al.*, « Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges », *in*: *Computers & Security* 28.*1-2* (Feb. 2009), pp. 18–28, ISSN: 01674048, DOI: 10.1016/j.cose.2008.08.003, URL: https://linkinghub.elsevier.com/retrieve/pii/S0167404808000692.

[GG20] Laze Gjorgiev and Sonja Gievska, « Time Series Anomaly Detection with Variational Autoencoder Using Mahalanobis Distance », *in*: *ICT Innovations 2020. Machine Learning and Applications*, ed. by Vesna Dimitrova and Ivica Dimitrovski, vol. 1316, Cham: Springer International Publishing, 2020, pp. 42–55, ISBN: 978-3-030-62097-4 978-3-030-62098-1, DOI: 10.1007/978-3-030-62098-1_4, URL: https://link.springer.com/10.1007/978-3-030-62098-1_4 (visited on 01/13/2023).

[Gho+07] Debanjan Ghosh *et al.*, « Self-Healing Systems — Survey and Synthesis », *in*: *Decision Support Systems* 42.*4* (Jan. 2007), pp. 2164–2185, ISSN: 01679236, DOI: 10.1016/j.dss.2006.06.011, URL: https://linkinghub.elsevier.com/retrieve/pii/S0167923606000807 (visited on 03/31/2022).

[GR22] Bimal Ghimire and Danda B. Rawat, « Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things », *in*: *IEEE Internet of Things Journal* 9.*11* (June 2022), pp. 8229–8249, ISSN: 2327-4662, DOI: 10.1109/JIOT.2022.3150363, URL: https://ieeexplore.ieee.org/abstract/document/9709603 (visited on 04/12/2024).

[Guo+23] Wei Guo *et al.*, « A New Federated Learning Model for Host Intrusion Detection System Under Non-IID Data », *in*: *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Oct. 2023, pp. 494–500, DOI: 10.1109/SMC53992.2023.10393972, URL: https://ieeexplore.ieee.org/document/10393972 (visited on 06/11/2024).

[Hab+17] Arash Habibi Lashkari *et al.*, « Characterization of Tor Traffic Using Time Based Features: » *in*: *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, 3rd International Conference on Information Systems Security and Privacy, Porto, Portugal: SCITEPRESS - Science and Technology Publications, 2017, pp. 253–262, ISBN: 978-989-758-209-7, DOI: 10.5220/0006105602530262, URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006105602530262 (visited on 10/15/2021).

[Had+08] Hamed Haddadi *et al.*, « Network Topologies: Inference, Modeling, and Generation », *in*: *IEEE Communications Surveys & Tutorials* 10.*2* (2008), pp. 48–69, ISSN: 1553-877X, DOI: 10.1109/COMST.2008.4564479, URL: https://ieeexplore.ieee.org/abstract/document/4564479 (visited on 07/07/2024).

[Hai+01] J W Haines *et al.*, « 1999 DARPA Intrusion Detection Evaluation: Design and Procedures », *in*: (2001), p. 188.

[Han+24] Weixiang Han *et al.*, « Heterogeneous Data-Aware Federated Learning for Intrusion Detection Systems via Meta-Sampling in Artificial Intelligence of Things », *in*: *IEEE Internet of Things Journal* 11.*8* (Apr. 2024), pp. 13340–13354, ISSN: 2327-4662, DOI: 10.1109/JIOT.2023.3337755, URL: https://ieeexplore.ieee.org/abstract/document/10334467 (visited on 04/12/2024).

[Har+17] Stephen Hardy *et al.*, « Private Federated Learning on Vertically Partitioned Data via Entity Resolution and Additively Homomorphic Encryption », Nov. 28, 2017, arXiv: 1711.10677 [cs], URL: http://arxiv.org/abs/1711.10677 (visited on 10/04/2021).

[HC18] David Hand and Peter Christen, « A Note on Using the F-measure for Evaluating Record Linkage Algorithms », *in*: *Statistics and Computing* 28.*3* (May 2018), pp. 539–547, ISSN: 0960-3174, 1573-1375, DOI: 10/gfw6dw, URL: http://link.springer.com/10.1007/s11222-017-9746-6 (visited on 11/10/2021).

[HDH23] Suli He, Chengwen Du, and M. Shamim Hossain, « 6G-enabled Consumer Electronics Device Intrusion Detection with Federated Meta-Learning and Digital Twins in a Meta-Verse Environment », *in*: *IEEE Transactions on Consumer Electronics* (2023), pp. 1–1, ISSN: 1558-4127, DOI: 10.1109/TCE.2023.3321846, URL: https://ieeexplore.ieee.org/abstract/document/10271257 (visited on 04/12/2024).

[He+24] Mingshu He, Xiaojuan Wang, *et al.*, « Reinforcement Learning Meets Network Intrusion Detection: A Transferable and Adaptable Framework for Anomaly Behavior Identification », *in*: *IEEE Transactions on Network and Service Management* 21.*2* (Apr. 2024), pp. 2477–2492, ISSN: 1932-4537, DOI: 10.1109/TNSM.2024.3352586, URL: https://ieeexplore.ieee.org/document/10399344 (visited on 06/30/2024).

[Hei+20] Xinhong Hei *et al.*, « A Trusted Feature Aggregator Federated Learning for Distributed Malicious Attack Detection », *in*: *Computers & Security* 99 (Dec. 2020), p. 102033, ISSN: 01674048, DOI: 10.1016/j.cose.2020.102033, URL: https://linkinghub.elsevier.com/retrieve/pii/S0167404820303060 (visited on 10/25/2021).

[Hel+22] Stijn Heldens *et al.*, « Litstudy: A Python Package for Literature Reviews », *in*: *SoftwareX* 20 (Dec. 1, 2022), p. 101207, ISSN: 2352-7110, DOI: 10.1016/j.softx.2022.101207, URL: https://www.sciencedirect.com/science/article/pii/S235271102200125X (visited on 06/06/2024).

[Hid18] Ochiai Hideya, *LAN-Security Monitoring Project*, Whitepaper, 2018, URL: https://lan-security.net/whitepaper.pdf (visited on 10/22/2021).

[Hua+21] Yutao Huang *et al.*, « Personalized Cross-Silo Federated Learning on Non-IID Data », *in*: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.*9* (May 18, 2021), pp. 7865–7873, ISSN: 2374-3468, 2159-5399, DOI: 10.1609/aaai.v35i9.16960, URL: https://ojs.aaai.org/index.php/AAAI/article/view/16960 (visited on 09/26/2022).

[Ism+24] Umar Audi Isma'ila *et al.*, « Review on Approaches of Federated Modeling in Anomaly-Based Intrusion Detection for IoT Devices », *in*: *IEEE Access* 12 (2024), pp. 30941–30961, ISSN: 2169-3536, DOI: 10.1109/ACCESS.2024.3369915, URL: https://ieeexplore.ieee.org/document/10445150 (visited on 04/24/2024).

[Jai+20] Abhinav Jain *et al.*, « Overview and Importance of Data Quality for Machine Learning Tasks », *in*: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, New York, NY, USA: Association for Computing Machinery, Aug. 20, 2020, pp. 3561–3562, ISBN: 978-1-4503-7998-4, DOI: 10.1145/3394486.3406477, URL: https://dl.acm.org/doi/10.1145/3394486.3406477 (visited on 03/28/2024).

[Jin+23] Dong Jin, Shuangwu Chen, *et al.*, « Federated Incremental Learning Based Evolvable Intrusion Detection System for Zero-Day Attacks », *in*: *IEEE Network* 37.*1* (Jan. 2023), pp. 125–132, ISSN: 1558-156X, DOI: 10.1109/MNET.018.2200349,

URL: https://ieeexplore.ieee.org/abstract/document/10110017 (visited on 04/12/2024).

[Jin+24] Zhigang Jin, Junyi Zhou, *et al.*, « FL-IIDS: A Novel Federated Learning-Based Incremental Intrusion Detection System », *in*: *Future Generation Computer Systems* 151 (Feb. 1, 2024), pp. 57–70, ISSN: 0167-739X, DOI: 10.1016/j.future.2023.09.019, URL: https://www.sciencedirect.com/science/article/pii/S0167739X23003503 (visited on 04/12/2024).

[Joh+16] Alistair E.W. Johnson *et al.*, « MIMIC-III, a Freely Accessible Critical Care Database », *in*: *Scientific Data* 3.*1* (May 24, 2016), p. 160035, ISSN: 2052-4463, DOI: 10.1038/sdata.2016.35, URL: https://doi.org/10.1038/sdata.2016.35.

[Kai+21] Peter Kairouz *et al.*, « Advances and Open Problems in Federated Learning », Mar. 8, 2021, arXiv: 1912.04977 [cs, stat], URL: http://arxiv.org/abs/1912.04977 (visited on 04/01/2022).

[Kan+20] Jiawen Kang *et al.*, « Reliable Federated Learning for Mobile Networks », *in*: *IEEE Wireless Communications* 27.*2* (Apr. 2020), pp. 72–80, ISSN: 1558-0687, DOI: 10.1109/MWC.001.1900119.

[Kar+20] Sai Praneeth Karimireddy, Satyen Kale, *et al.*, « SCAFFOLD: Stochastic Controlled Averaging for Federated Learning », *in*: *Proceedings of the 37th International Conference on Machine Learning*, International Conference on Machine Learning, PMLR, Nov. 21, 2020, pp. 5132–5143, URL: https://proceedings.mlr.press/v119/karimireddy20a.html (visited on 06/23/2024).

[KC03] J.O. Kephart and D.M. Chess, « The Vision of Autonomic Computing », *in*: *Computer* 36.*1* (Jan. 2003), pp. 41–50, ISSN: 0018-9162, DOI: 10.1109/MC.2003.1160055, URL: http://ieeexplore.ieee.org/document/1160055/.

[KC07] B. Kitchenham and S Charters, *Guidelines for Performing Systematic Literature Reviews in Software Engineering*, EBSE-2007-01, 2007.

[Kes07] S. Keshav, « How to Read a Paper », *in*: *acm special interest group on data communication* 37.*3* (2007), pp. 83–84.

[KGS21] Muah Kim, Onur Gunlu, and Rafael F Schaefer, « Federated Learning with Local Differential Privacy: Trade-offs between Privacy, Utility, and Communication », *in*: (2021).

[KHJ21] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi, « Learning from History for Byzantine Robust Optimization », *in*: *Proceedings of the 38th International Conference on Machine Learning*, International Conference on Machine Learning, PMLR, July 1, 2021, pp. 5311–5319, URL: https://proceedings.mlr.press/v139/karimireddy21a.html (visited on 10/21/2022).

[Kho+21] Tran Viet Khoa *et al.*, « Deep Transfer Learning: A Novel Collaborative Learning Model for Cyberattack Detection Systems in IoT Networks », Dec. 2, 2021, arXiv: 2112.00988 [cs], URL: http://arxiv.org/abs/2112.00988 (visited on 01/31/2022).

[Kim+20] Seongwoo Kim, He Cai, *et al.*, « Collaborative Anomaly Detection for Internet of Things Based on Federated Learning », *in*: *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, 2020 IEEE/CIC International Conference on Communications in China (ICCC), Chongqing, China: IEEE, Aug. 9, 2020, pp. 623–628, ISBN: 978-1-72817-327-6, DOI: 10.1109/ICCC49849.2020.9238913, URL: https://ieeexplore.ieee.org/document/9238913/ (visited on 10/25/2021).

[Kol+16] Constantinos Kolias *et al.*, « Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset », *in*: *IEEE Communications Surveys & Tutorials* 18.*1* (2016), pp. 184–208, ISSN: 1553-877X, DOI: 10.1109/COMST.2015.2402161, URL: http://ieeexplore.ieee.org/document/7041170/.

[Kon+16a] Jakub Konený, H. Brendan McMahan, Daniel Ramage, *et al.*, « Federated Optimization: Distributed Machine Learning for On-Device Intelligence », *in*: (Oct. 8, 2016), pp. 1–38, URL: http://arxiv.org/abs/1610.02527.

[Kon+16b] Jakub Konený, H. Brendan McMahan, Felix X. Yu, *et al.*, « Federated Learning: Strategies for Improving Communication Efficiency », *in*: (Oct. 18, 2016), pp. 1–10, URL: http://arxiv.org/abs/1610.05492.

[Kor+19] Nickolaos Koroniotis *et al.*, « Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset », *in*: *Future Generation Computer Systems* 100 (Nov. 2019), pp. 779–796, ISSN: 0167739X, DOI: 10.1016/j.future.2019.05.041, URL: https://linkinghub.elsevier.com/retrieve/pii/S0167739X18327687 (visited on 10/23/2021).

[Kun+22] Achintya Kundu *et al.*, « Robustness and Personalization in Federated Learning: A Unified Approach via Regularization », *in*: *2022 IEEE International Conference on Edge Computing and Communications (EDGE)*, 2022 IEEE International Conference on Edge Computing and Communications (EDGE), July 2022, pp. 1–11, DOI: 10.1109/EDGE55608.2022.00014, URL: https://ieeexplore.ieee.org/document/9860349 (visited on 07/01/2024).

[Lan+22] Maxime Lanvin *et al.*, « Errors in the CICIDS2017 Dataset and the Significant Differences in Detection Performances It Makes », *in*: *CRiSIS 2022 - International Conference on Risks and Security of Internet and Systems*, Sousse, Tunisia, Dec. 2022, pp. 1–16, URL: https://hal.archives-ouvertes.fr/hal-03775466 (visited on 09/29/2022).

[Lau+17] Andrés Laurito *et al.*, « TopoGen: A Network Topology Generation Architecture with Application to Automating Simulations of Software Defined Networks », *in*: *2017 Winter Simulation Conference (WSC)*, 2017 Winter Simulation Conference

(WSC), Dec. 2017, pp. 1049–1060, DOI: 10.1109/WSC.2017.8247854, URL: https://ieeexplore.ieee.org/abstract/document/8247854 (visited on 07/07/2024).

[Lav+22a] Leo Lavaur, Benjamin Coste, *et al.*, « Federated Learning as Enabler for Collaborative Security between Not Fully-Trusting Distributed Parties », *in*: *Proceedings of the 29th Computer & Electronics Security Application Rendezvous (C&ESAR): Ensuring Trust in a Decentralized World*, 2022, pp. 65–80, URL: http://ceur-ws.org/Vol-3329/paper-04.pdf.

[Lav+22b] Leo Lavaur, Marc-Oliver Pahl, *et al.*, « The Evolution of Federated Learning-based Intrusion Detection and Mitigation: A Survey », *in*: *IEEE Transactions on Network and Service Management*, Special Issue on Network Security Management (June 2022).

[LBA24a] **Léo Lavaur**, Yann Busnel, and Fabien Autrel, « Demo: Highlighting the Limits of Federated Learning in Intrusion Detection », *in*: *Proceedings of the 44th International Conference on Distributed Computing Systems (ICDCS)*, Jersey City, NJ, USA, July 2024.

[LBA24b] **Léo Lavaur**, Yann Busnel, and Fabien Autrel, « Systematic Analysis of Label-flipping Attacks against Federated Learning in Collaborative Intrusion Detection Systems », *in*: *Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES), Workshop on Behavioral Authentication for System Security (BASS)*, Vienna, Austria, Aug. 2024.

[LDR19] Eliot Lear, Ralph Droms, and Dan Romascanu, *Manufacturer Usage Description Specification*, RFC 8520, RFC Editor, Mar. 2019, DOI: 10.17487/RFC8520, URL: https://rfc-editor.org/rfc/rfc8520.txt.

[Lec+98] Y. Lecun *et al.*, « Gradient-Based Learning Applied to Document Recognition », *in*: *Proceedings of the IEEE* 86.*11* (Nov. 1998), pp. 2278–2324, ISSN: 1558-2256, DOI: 10.1109/5.726791, URL: https://ieeexplore.ieee.org/document/726791 (visited on 11/09/2023).

[Léo+22] **Léo Lavaur**, Benjamin Coste, *et al.*, « Federated Learning as Enabler for Collaborative Security between Not Fully-Trusting Distributed Parties », *in*: *Proceedings of the 29th Computer & Electronics Security Application Rendezvous (C&ESAR)*, Rennes, France, Oct. 2022.

[Léo+24] **Léo Lavaur**, Pierre-Marie Lechevalier, *et al.*, « RADAR: Model Quality Assessment for Reputation-aware Collaborative Federated Learning », *in*: *Proceedings of the 43rd International Symposium on Reliable Distributed Systems (SRDS)*, Charlotte, NC, USA, Sept. 2024.

[Li+20a] Beibei Li, Yuhao Wu, *et al.*, « DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber-Physical Systems », *in*: *IEEE Transactions on Industrial Informatics* 3203.*c* (2020), pp. 1–1, ISSN: 1551-3203, DOI: 10.1109/TII.2020.3023430, URL: https://ieeexplore.ieee.org/document/9195012/.

[Li+20b] Kun Li, Huachun Zhou, *et al.*, « Distributed Network Intrusion Detection System in Satellite-Terrestrial Integrated Networks Using Federated Learning », *in*: *IEEE Access* 8 (2020), pp. 214852–214865, ISSN: 2169-3536, DOI: 10.1109/ACCESS.2020.3041641, URL: https://ieeexplore.ieee.org/document/9274426/ (visited on 10/25/2021).

[Li+20c] Tian Li, Anit Kumar Sahu, *et al.*, « Federated Optimization in Heterogeneous Networks », Apr. 21, 2020, arXiv: 1812.06127 [cs, stat], URL: http://arxiv.org/abs/1812.06127 (visited on 09/20/2021).

[Liu+20] Lumin Liu, Jun Zhang, S.H. Song, *et al.*, « Client-Edge-Cloud Hierarchical Federated Learning », *in*: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, ICC 2020 - 2020 IEEE International Conference on Communications (ICC), June 2020, pp. 1–6, DOI: 10.1109/ICC40277.2020.9148862, URL: https://ieeexplore.ieee.org/document/9148862 (visited on 06/23/2024).

[Liu+21] Hong Liu, Shuaipeng Zhang, Pengfei Zhang, *et al.*, « Blockchain and Federated Learning for Collaborative Intrusion Detection in Vehicular Edge Computing », *in*: *IEEE Transactions on Vehicular Technology* 70.*6* (June 2021), pp. 6073–6084, ISSN: 0018-9545, 1939-9359, DOI: 10.1109/TVT.2021.3076780, URL: https://ieeexplore.ieee.org/document/9420262/ (visited on 10/04/2021).

[LL19] Hongyu Liu and Bo Lang, « Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey », *in*: *Applied Sciences* 9.*20* (Oct. 17, 2019), p. 4396, ISSN: 2076-3417, DOI: 10.3390/app9204396, URL: https://www.mdpi.com/2076-3417/9/20/4396 (visited on 03/11/2022).

[LMK22] Wenjuan Li, Weizhi Meng, and Lam For Kwok, « Surveying Trust-Based Collaborative Intrusion Detection: State-of-the-Art, Challenges and Future Directions », *in*: *IEEE Communications Surveys & Tutorials* 24.*1* (2022), pp. 280–305, ISSN: 1553-877X, DOI: 10.1109/COMST.2021.3139052, URL: https://ieeexplore.ieee.org/document/9663537 (visited on 06/22/2024).

[Lo+21] Sin Kit Lo *et al.*, « A Systematic Literature Review on Federated Machine Learning: From A Software Engineering Perspective », *in*: *ACM Computing Surveys* 54.*5* (June 2021), pp. 1–39, ISSN: 0360-0300, 1557-7341, DOI: 10.1145/3450288, arXiv: 2007.11354, URL: http://arxiv.org/abs/2007.11354 (visited on 10/04/2021).

[LP22] Siamak Layeghy and Marius Portmann, *On Generalisability of Machine Learning-based Network Intrusion Detection Systems*, May 9, 2022, arXiv: 2205.04112 [cs], URL: http://arxiv.org/abs/2205.04112 (visited on 03/23/2023), pre-published.

[LYY20] Lingjuan Lyu, Han Yu, and Qiang Yang, « Threats to Federated Learning: A Survey », *in*: *arXiv* (Mar. 4, 2020), URL: http://arxiv.org/abs/2003.02133.

[Ma+22] Zhuoran Ma *et al.*, « ShieldFL: Mitigating Model Poisoning Attacks in Privacy-Preserving Federated Learning », *in*: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 1639–1654, ISSN: 1556-6013, 1556-6021, DOI: 10.1109/TIFS.2022.3169918, URL: https://ieeexplore.ieee.org/document/9762272/ (visited on 07/05/2022).

[Mar+19] Samuel Marchal *et al.*, « AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication », *in*: *IEEE Journal on Selected Areas in Communications* 37.*6* (June 2019), pp. 1402–1412, ISSN: 0733-8716, 1558-0008, DOI: 10.1109/JSAC.2019.2904364, URL: https://ieeexplore.ieee.org/document/8664655/ (visited on 06/04/2021).

[McM+17] Brendan McMahan *et al.*, « Communication-Efficient Learning of Deep Networks from Decentralized Data », *in*: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ed. by Aarti Singh and Jerry Zhu, vol. 54, Proceedings of Machine Learning Research, PMLR, Apr. 20–22, 2017, pp. 1273–1282, URL: https://proceedings.mlr.press/v54/mcmahan17a.html.

[Med+01] A. Medina *et al.*, « BRITE: An Approach to Universal Topology Generation », *in*: *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Aug. 2001, pp. 346–353, DOI: 10.1109/MASCOT.2001.948886.

[Mei+18] Yair Meidan *et al.*, « N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders », *in*: *IEEE Pervasive Computing* 17.*3* (July 2018), pp. 12–22, ISSN: 1536-1268, 1558-2590, DOI: 10.1109/MPRV.2018.03367731, arXiv: 1805.03409, URL: http://arxiv.org/abs/1805.03409 (visited on 10/23/2021).

[Men+15] Guozhu Meng, Yang Liu, *et al.*, « Collaborative Security: A Survey and Taxonomy », *in*: *ACM Computing Surveys* 48.*1* (July 22, 2015), 1:1–1:42, ISSN: 0360-0300, DOI: 10.1145/2785733, URL: https://doi.org/10.1145/2785733 (visited on 06/22/2024).

[Men+18] Weizhi Meng, Elmar Wolfgang Tischhauser, *et al.*, « When Intrusion Detection Meets Blockchain Technology: A Review », *in*: *IEEE Access* 6 (2018), pp. 10179–10188, ISSN: 2169-3536, DOI: 10.1109/ACCESS.2018.2799854, URL: http://ieeexplore.ieee.org/document/8274922/.

[Mer+23] Mohamed Amine Merzouk *et al.*, « Parameterizing Poisoning Attacks in Federated Learning-Based Intrusion Detection », *in*: *Proceedings of the 18th International Conference on Availability, Reliability and Security*, ARES '23, New York, NY, USA: Association for Computing Machinery, Aug. 29, 2023, pp. 1–8, ISBN: 9798400707728, DOI: 10.1145/3600160.3605090, URL: https://dl.acm.org/doi/10.1145/3600160.3605090 (visited on 01/29/2024).

[MG14] Thomas Morris and Wei Gao, « Industrial Control System Traffic Data Sets for Intrusion Detection Research », *in*: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, ed. by Eduardo Bayro-Corrochano and Edwin Hancock, vol. 8827, Cham: Springer International Publishing, 2014, pp. 65–78, ISBN: 978-3-319-12567-1 978-3-319-12568-8, DOI: 10.1007/978-3-662-45355-1_5, URL: http://link.springer.com/10.1007/978-3-662-45355-1_5 (visited on 06/10/2021).

[ML15] Stuart Murdoch and Nick Leaver, « Anonymity vs. Trust in Cyber-Security Collaboration », *in*: *Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security*, New York, NY, USA: ACM, Oct. 12, 2015, pp. 27–29, ISBN: 978-1-4503-3822-6, DOI: 10.1145/2808128.2808134, URL: https://dl.acm.org/doi/10.1145/2808128.2808134.

[Mot+21a] Viraaji Mothukuri, Prachi Khare, *et al.*, « Federated Learning-based Anomaly Detection for IoT Security Attacks », *in*: *IEEE Internet of Things Journal* (2021), pp. 1–1, ISSN: 2327-4662, DOI: 10/gmhhmw.

[Mot+21b] Viraaji Mothukuri, Reza M. Parizi, *et al.*, « A Survey on Security and Privacy of Federated Learning », *in*: *Future Generation Computer Systems* 115 (Feb. 2021), pp. 619–640, ISSN: 0167739X, DOI: 10.1016/j.future.2020.10.007, URL: https://doi.org/10.1016/j.future.2020.10.007.

[Mou+20] Nour Moustafa, Marwa Keshky, *et al.*, « Federated TON_IoT Windows Datasets for Evaluating AI-Based Security Applications », *in*: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Dec. 2020, pp. 848–855, DOI: 10.1109/TrustCom50675.2020.00114.

[Mou21] Nour Moustafa, « A New Distributed Architecture for Evaluating AI-based Security Systems at the Edge: Network TON_IoT Datasets », *in*: *Sustainable Cities and Society* 72 (Sept. 1, 2021), p. 102994, ISSN: 2210-6707, DOI: 10.1016/j.scs.2021.102994, URL: https://www.sciencedirect.com/science/article/pii/S2210670721002808 (visited on 06/21/2024).

[MS15] Nour Moustafa and Jill Slay, « UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set) », *in*: *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015 Military Communications and Information Systems Conference (MilCIS), Nov. 2015, pp. 1–6, DOI: 10.1109/MilCIS.2015.7348942, URL: https://ieeexplore.ieee.org/abstract/document/7348942 (visited on 10/09/2023).

[Nas+22] Mohammad Naseri *et al.*, « Cerberus: Exploring Federated Prediction of Security Events », *in*: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, New York, NY, USA: Association for Computing Machinery, Nov. 7, 2022, pp. 2337–2351, ISBN: 978-1-4503-9450-5, DOI:

10.1145/3548606.3560580, URL: https://doi.org/10.1145/3548606.3560580 (visited on 11/20/2023).

[Nat24] National Institute of Standards and Technology, *The NIST Cybersecurity Framework (CSF) 2.0*, NIST CSWP 29, Gaithersburg, MD: National Institute of Standards and Technology, Feb. 26, 2024, NIST CSWP 29, DOI: 10.6028/NIST.CSWP.29, URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.29.pdf (visited on 05/23/2024).

[NDG22] Evgenia Novikova, Elena Doynikova, and Sergey Golubev, « Federated Learning for Intrusion Detection in the Critical Infrastructures: Vertically Partitioned Data Use Case », *in*: *Algorithms* 15.*4* (4 Mar. 23, 2022), p. 104, ISSN: 1999-4893, DOI: 10.3390/a15040104, URL: https://www.mdpi.com/1999-4893/15/4/104 (visited on 07/05/2022).

[Ngu+19] Thien Duc Nguyen, Samuel Marchal, *et al.*, « DÏoT: A Federated Self-learning Anomaly Detection System for IoT », *in*: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, vol. 2019-July, IEEE, July 2019, pp. 756–767, ISBN: 978-1-72812-519-0, DOI: 10.1109/ICDCS.2019.00080, URL: https://ieeexplore.ieee.org/document/8884802/.

[Ngu+20] Thien Duc Nguyen, Phillip Rieger, Markus Miettinen, *et al.*, « Poisoning Attacks on Federated Learning-based IoT Intrusion Detection System », *in*: *Proceedings 2020 Workshop on Decentralized IoT Systems and Security*, Workshop on Decentralized IoT Systems and Security, San Diego, CA: Internet Society, 2020, ISBN: 978-1-891562-64-8, DOI: 10.14722/diss.2020.23003, URL: https://www.ndss-symposium.org/wp-content/uploads/2020/04/diss2020-23003-paper.pdf (visited on 01/29/2024).

[Ngu+21] Thien Duc Nguyen, Phillip Rieger, Hossein Yalame, *et al.*, « FLGUARD: Secure and Private Federated Learning », Jan. 21, 2021, arXiv: 2101.02281 [cs], URL: http://arxiv.org/abs/2101.02281 (visited on 05/18/2021).

[Ngu+22] Thien Duc Nguyen, Phillip Rieger, Huili Chen, *et al.*, « FLAME: Taming Backdoors in Federated Learning », *in*: 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 1415–1432, ISBN: 978-1-939133-31-1, URL: https://www.usenix.org/conference/usenixsecurity22/presentation/nguyen (visited on 03/06/2024).

[NM22] Florian Nuding and Rudolf Mayer, « Data Poisoning in Sequential and Parallel Federated Learning », *in*: *Proceedings of the 2022 ACM on International Workshop on Security and Privacy Analytics*, CODASPY '22: Twelveth ACM Conference on Data and Application Security and Privacy, Baltimore MD USA: ACM, Apr. 18, 2022, pp. 24–34, ISBN: 978-1-4503-9230-3, DOI: 10.1145/3510548.3519372, URL: https://dl.acm.org/doi/10.1145/3510548.3519372 (visited on 07/05/2022).

[Ouy+22] Xiaomin Ouyang *et al.*, « ClusterFL: A Clustering-based Federated Learning System for Human Activity Recognition », *in*: *ACM Transactions on Sensor Networks* 19.*1* (Dec. 8, 2022), 17:1–17:32, ISSN: 1550-4859, DOI: 10.1145/3554980, URL: https://dl.acm.org/doi/10.1145/3554980 (visited on 01/12/2024).

[OWN21] Yazan Otoum, Yue Wan, and Amiya Nayak, « Federated Transfer Learning-Based IDS for the Internet of Medical Things (IoMT) », *in*: *2021 IEEE Globecom Workshops (GC Wkshps)*, 2021 IEEE Globecom Workshops (GC Wkshps), Madrid, Spain: IEEE, Dec. 2021, pp. 1–6, ISBN: 978-1-66542-390-8, DOI: 10/gpbg4z, URL: https://ieeexplore.ieee.org/document/9682118/ (visited on 01/31/2022).

[PA18] Marc-Oliver Pahl and Francois Xavier Aubet, « All Eyes on You: Distributed Multi-Dimensional IoT Microservice Anomaly Detection », *in*: *14th International Conference on Network and Service Management, CNSM 2018, 1st Workshop on Segment Routing and Service Function Chaining* (2018), pp. 72–80.

[Pai99] Pascal Paillier, « Public-Key Cryptosystems Based on Composite Degree Residuosity Classes », *in*: *Advances in Cryptology — EUROCRYPT '99*, ed. by Jacques Stern, vol. 1592, Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238, ISBN: 978-3-540-65889-4, DOI: 10.1007/3-540-48910-X_16, URL: http://link.springer.com/10.1007/3-540-48910-X_16 (visited on 07/05/2021).

[Pan+22] Dinelka Panagoda *et al.*, « Application of Federated Learning in Health Care Sector for Malware Detection and Mitigation Using Software Defined Networking Approach », *in*: *2022 2nd Asian Conference on Innovation in Technology (ASIANCON)*, 2022 2nd Asian Conference on Innovation in Technology (ASIANCON), Aug. 2022, pp. 1–6, DOI: 10.1109/ASIANCON55314.2022.9909488, URL: https://ieeexplore.ieee.org/abstract/document/9909488 (visited on 04/12/2024).

[PB23] Balázs Pejó and Gergely Biczók, « Quality Inference in Federated Learning With Secure Aggregation », *in*: *IEEE Transactions on Big Data* 9.*5* (Oct. 2023), pp. 1430–1437, ISSN: 2332-7790, DOI: 10.1109/TBDATA.2023.3280406, URL: https://ieeexplore.ieee.org/abstract/document/10138056 (visited on 03/27/2024).

[Per+20] Neehar Peri *et al.*, « Deep K-NN Defense Against Clean-Label Data Poisoning Attacks », *in*: *Computer Vision – ECCV 2020 Workshops*, ed. by Adrien Bartoli and Andrea Fusiello, Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 55–70, ISBN: 978-3-030-66415-2, DOI: 10.1007/978-3-030-66415-2_4.

[PG22] Trung V. Phan and Tri Gia Nguyen, « FEAR: Federated Cyber-Attack Reaction in Distributed Software-Defined Networks with Deep Q-Network », *in*: *2022 Wireless Telecommunications Symposium (WTS)*, 2022 Wireless Telecommunications Symposium (WTS), Apr. 2022, pp. 1–7, DOI: 10.1109/WTS53620.2022.9768169.

[Pop+21a] Segun I. Popoola, Ruth Ande, *et al.*, « Federated Deep Learning for Zero-Day Botnet Attack Detection in IoT Edge Devices », *in*: *IEEE Internet of Things Journal* (2021), pp. 1–1, ISSN: 2327-4662, 2372-2541, DOI: 10.1109/JIOT.2021.3100755, URL: https://ieeexplore.ieee.org/document/9499122/ (visited on 10/01/2021).

[Pop+21b] Segun I. Popoola, Guan Gui, *et al.*, « Federated Deep Learning for Collaborative Intrusion Detection in Heterogeneous Networks », *in*: *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), Sept. 2021, pp. 1–6, DOI: 10.1109/VTC2021-Fall52928.2021.9625505.

[Pop+22] Segun Popoola, Bamidele Adebisi, *et al.*, *Optimizing Deep Learning Model Hyperparameters for Botnet Attack Detection in IoT Networks*, preprint, Apr. 8, 2022, DOI: 10.36227/techrxiv.19501885.v1, URL: https://www.techrxiv.org/articles/preprint/Optimizing_Deep_Learning_Model_Hyperparameters_for_Botnet_Attack_Detection_in_IoT_Networks/19501885/1 (visited on 04/14/2022).

[Pop+23] Segun I. Popoola, Agbotiname L. Imoize, *et al.*, « Federated Deep Learning for Intrusion Detection in Consumer-Centric Internet of Things », *in*: *IEEE Transactions on Consumer Electronics* (2023), pp. 1–1, ISSN: 1558-4127, DOI: 10.1109/TCE.2023.3347170, URL: https://ieeexplore.ieee.org/abstract/document/10373897 (visited on 04/12/2024).

[PZ19] Ali Pala and Jun Zhuang, « Information Sharing in Cybersecurity: A Review », *in*: *Decision Analysis* 16.*3* (Sept. 2019), pp. 172–196, ISSN: 1545-8490, DOI: 10.1287/deca.2018.0387, URL: http://pubsonline.informs.org/doi/10.1287/deca.2018.0387.

[Qin+20a] Qiaofeng Qin, Konstantinos Poularakis, *et al.*, « Line-Speed and Scalable Intrusion Detection at the Network Edge via Federated Learning », *in*: *2020 IFIP Networking Conference (Networking)*, 2020 IFIP Networking Conference (Networking), June 2020, pp. 352–360, URL: https://ieeexplore.ieee.org/document/9142704 (visited on 04/12/2024).

[Qin+20b] Qiaofeng Qin, Konstantinos Poularakis, *et al.*, « Line-Speed and Scalable Intrusion Detection at the Network Edge via Federated Learning », *in*: (June 2020), p. 9.

[QK21] Yang Qin and Masaaki Kondo, « Federated Learning-Based Network Intrusion Detection with a Feature Selection Approach », *in*: *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, 2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Kuala Lumpur, Malaysia: IEEE, June 12, 2021, pp. 1–6, ISBN: 978-1-66543-897-1, DOI: 10.1109/ICECCE52056.2021.9514222, URL: https://ieeexplore.ieee.org/document/9514222/ (visited on 10/04/2021).

[Quy+22] Nguyen Huu Quyen *et al.*, « Federated Intrusion Detection on Non-IID Data for IIoT Networks Using Generative Adversarial Networks and Reinforcement Learning », *in*: *Information Security Practice and Experience*, ed. by Chunhua Su, Dimitris Gritzalis, and Vincenzo Piuri, Cham: Springer International Publishing, 2022, pp. 364–381, ISBN: 978-3-031-21280-2, DOI: 10.1007/978-3-031-21280-2_20.

[Rah+20] Sawsan Abdul Rahman *et al.*, « Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning? », *in*: *IEEE Network* 34.*6* (Nov. 2020), pp. 310–317, ISSN: 0890-8044, 1558-156X, DOI: 10.1109/MNET.011.2000286, URL: https://ieeexplore.ieee.org/document/9183799/ (visited on 06/01/2021).

[Res+00] Paul Resnick *et al.*, « Reputation Systems », *in*: *Communications of the ACM* 43.*12* (Dec. 1, 2000), pp. 45–48, ISSN: 0001-0782, DOI: 10.1145/355112.355122, URL: https://doi.org/10.1145/355112.355122 (visited on 02/01/2023).

[Rin+17a] Markus Ring, Sarah Wunderlich, Dominik Grüdl, *et al.*, « Creation of Flow-Based Data Sets for Intrusion Detection », *in*: *Journal of Information Warfare* 16.*4* (2017), pp. 41–54, ISSN: 14453312, 14453347, JSTOR: 26504117, URL: https://www.jstor.org/stable/26504117.

[Rin+17b] Markus Ring, Sarah Wunderlich, Dominik Grüdl, *et al.*, « Flow-Based Benchmark Data Sets for Intrusion Detection », *in*: *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)* (2017), pp. 361–369, ISSN: 20488610.

[Rin+19] Markus Ring, Sarah Wunderlich, Deniz Scheuring, *et al.*, « A Survey of Network-Based Intrusion Detection Data Sets », *in*: *Computers & Security* 86 (Sept. 1, 2019), pp. 147–167, ISSN: 0167-4048, DOI: 10.1016/j.cose.2019.06.005, URL: https://www.sciencedirect.com/science/article/pii/S016740481930118X (visited on 07/07/2024).

[RN21] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Fourth Edition, Pearson Series in Artificial Intelligence, Hoboken, NJ: Pearson, 2021, 1115 pp., ISBN: 978-0-13-461099-3.

[Rod+23] Nuria Rodríguez-Barroso *et al.*, « Survey on Federated Learning Threats: Concepts, Taxonomy on Attacks and Defences, Experimental Study and Challenges », *in*: *Information Fusion* 90 (Feb. 1, 2023), pp. 148–173, ISSN: 1566-2535, DOI: 10.1016/j.inffus.2022.09.011, URL: https://www.sciencedirect.com/science/article/pii/S1566253522001439 (visited on 09/29/2023).

[RWP19] Shailendra Rathore, Byung Wook Kwon, and Jong Hyuk Park, « BlockSecIoTNet: Blockchain-based Decentralized Security Architecture for IoT Network », *in*: *Journal of Network and Computer Applications* 143 (December 2018 Oct. 2019), pp. 167–177, ISSN: 10848045, DOI: 10.1016/j.jnca.2019.06.019, URL: https://doi.org/10.1016/j.jnca.2019.06.019.

[SA21] Sudipan Saha and Tahir Ahmad, « Federated Transfer Learning: Concept and Applications », Mar. 6, 2021, arXiv: 2010.15561 [cs], URL: http://arxiv.org/abs/2010.15561 (visited on 10/04/2021).

[SB20] Jagdeep Singh and Sunny Behal, « Detection and Mitigation of DDoS Attacks in SDN: A Comprehensive Review, Research Challenges and Future Directions », *in*: *Computer Science Review* 37 (Aug. 2020), p. 100279, ISSN: 15740137, DOI: 10.1016/j.cosrev.2020.100279, URL: https://linkinghub.elsevier.com/retrieve/pii/S1574013720301647 (visited on 04/09/2022).

[SEO21] Yuwei Sun, Hiroshi Esaki, and Hideya Ochiai, « Adaptive Intrusion Detection in the Networking of Large-Scale LANs With Segmented Federated Learning », *in*: *IEEE Open Journal of the Communications Society* 2 (2021), pp. 102–112, ISSN: 2644-125X, DOI: 10.1109/OJCOMS.2020.3044323, URL: https://ieeexplore.ieee.org/document/9296578/ (visited on 10/04/2021).

[Sha+24] Yao Shan *et al.*, « CFL-IDS: An Effective Clustered Federated Learning Framework for Industrial Internet of Things Intrusion Detection », *in*: *IEEE Internet of Things Journal* 11.*6* (Mar. 2024), pp. 10007–10019, ISSN: 2327-4662, DOI: 10.1109/JIOT.2023.3324302, URL: https://ieeexplore.ieee.org/abstract/document/10285326 (visited on 04/12/2024).

[She+20] Sheng Shen, Tianqing Zhu, *et al.*, « From Distributed Machine Learning To Federated Learning: In The View Of Data Privacy And Security », *in*: *Concurrency and Computation: Practice and Experience* (Sept. 23, 2020), cpe.6002, ISSN: 1532-0626, 1532-0634, DOI: 10.1002/cpe.6002, arXiv: 2010.09258, URL: http://arxiv.org/abs/2010.09258 (visited on 10/04/2021).

[SHG18] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, « Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization », *in*: *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 4th International Conference on Information Systems Security and Privacy, Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 108–116, ISBN: 978-989-758-282-0, DOI: 10.5220/0006639801080116, URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006639801080116 (visited on 10/14/2021).

[Sig99] SigKDD, *KDD Cup 1999 Dataset*, 1999, URL: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (visited on 04/26/2021).

[SLP21] Mohanad Sarhan, Siamak Layeghy, and Marius Portmann, *Towards a Standard Feature Set for Network Intrusion Detection System Datasets*, May 14, 2021, arXiv: 2101.11315 [cs], URL: http://arxiv.org/abs/2101.11315 (visited on 09/12/2022), pre-published.

[SLP22] Mohanad Sarhan, Siamak Layeghy, and Marius Portmann, « Towards a Standard Feature Set for Network Intrusion Detection System Datasets », *in*: *Mobile Networks and Applications* 27.*1* (Feb. 1, 2022), pp. 357–370, ISSN: 1572-8153, DOI: 10.1007/s11036-021-01843-0, URL: https://doi.org/10.1007/s11036-021-01843-0 (visited on 04/23/2024).

[Sna+92] Steven R. Snapp *et al.*, « The DIDS (Distributed Intrusion Detection System) Prototype », *in*: *USENIX Summer 1992 Technical Conference (USENIX Summer 1992 Technical Conference)*, San Antonio, TX: USENIX Association, June 1992, URL: https://www.usenix.org/conference/usenix-summer-1992-technical-conference/dids-distributed-intrusion-detection-system.

[SOE20] Yuwei Sun, Hideya Ochiai, and Hiroshi Esaki, « Intrusion Detection with Segmented Federated Learning for Large-Scale Multiple LANs », *in*: *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, United Kingdom: IEEE, July 2020, pp. 1–8, ISBN: 978-1-72816-926-2, DOI: 10.1109/IJCNN48605.2020.9207094, URL: https://ieeexplore.ieee.org/document/9207094/ (visited on 10/01/2021).

[Son+22] Zhendong Song *et al.*, « Reputation-Based Federated Learning for Secure Wireless Networks », *in*: *IEEE Internet of Things Journal* 9.*2* (Jan. 2022), pp. 1212–1226, ISSN: 2327-4662, DOI: 10.1109/JIOT.2021.3079104.

[SSF16] Florian Skopik, Giuseppe Settanni, and Roman Fiedler, « A Problem Shared Is a Problem Halved: A Survey on the Dimensions of Collective Cyber Defense through Security Information Sharing », *in*: *Computers & Security* 60 (2016), pp. 154–176, ISSN: 01674048, DOI: 10.1016/j.cose.2016.04.003.

[ST19] William Schneble and Geethapriya Thamilarasu, « Attack Detection Using Federated Learning in Medical Cyber-Physical Systems », *in*: *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, International Conference on Computer Communications and Networks, Aug. 2019.

[STS16] Shiqi Shen, Shruti Tople, and Prateek Saxena, « Auror: Defending against Poisoning Attacks in Collaborative Deep Learning Systems », *in*: *Proceedings of the 32nd Annual Conference on Computer Security Applications*, New York, NY, USA: ACM, Dec. 5, 2016, pp. 508–519, ISBN: 978-1-4503-4771-6, DOI: 10.1145/2991079.2991125, URL: https://dl.acm.org/doi/10.1145/2991079.2991125.

[Sun+20] Wen Sun, Shiyu Lei, *et al.*, « Adaptive Federated Learning and Digital Twin for Industrial Internet of Things », Oct. 31, 2020, arXiv: 2010.13058 [cs], URL: http://arxiv.org/abs/2010.13058 (visited on 10/04/2021).

[Sun+22] Gan Sun, Yang Cong, *et al.*, « Data Poisoning Attacks on Federated Machine Learning », *in*: *IEEE Internet of Things Journal* 9.*13* (July 2022), pp. 11365–11375, ISSN: 2327-4662, DOI: 10.1109/JIOT.2021.3128646, URL: https://ieeexplore.ieee.org/abstract/document/9618642 (visited on 12/06/2023).

[Tan+02] Hongsuda Tangmunarunkit *et al.*, « Network Topology Generators: Degree-Based vs. Structural », *in*: *SIGCOMM Comput. Commun. Rev.* 32.*4* (Aug. 19, 2002), pp. 147–159, ISSN: 0146-4833, DOI: 10.1145/964725.633040, URL: https://doi.org/10.1145/964725.633040 (visited on 07/07/2024).

[Tan+22] Xavier Tan *et al.*, « Reputation-Aware Federated Learning Client Selection Based on Stochastic Integer Programming », *in*: *IEEE Transactions on Big Data* (2022), pp. 1–12, ISSN: 2332-7790, DOI: 10.1109/TBDATA.2022.3191332.

[Tan+23] Zhenheng Tang *et al.*, « GossipFL: A Decentralized Federated Learning Framework With Sparsified and Adaptive Communication », *in*: *IEEE Transactions on Parallel and Distributed Systems* 34.*3* (Mar. 2023), pp. 909–922, ISSN: 1558-2183, DOI: 10.1109/TPDS.2022.3230938, URL: https://ieeexplore.ieee.org/document/9996127 (visited on 06/23/2024).

[Tav+09] Mahbod Tavallaee *et al.*, « A Detailed Analysis of the KDD CUP 99 Data Set », *in*: *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, IEEE, July 2009, pp. 1–6, ISBN: 978-1-4244-3763-4, DOI: 10.1109/CISDA.2009.5356528, URL: http://ieeexplore.ieee.org/document/5356528/.

[The] The Open Information Security Foundation, *Suricata IDS*, URL: https://suricata.io/ (visited on 06/30/2024).

[Thi+22] Huynh Thai Thi *et al.*, « Federated Learning-Based Cyber Threat Hunting for APT Attack Detection in SDN-Enabled Networks », *in*: *2022 21st International Symposium on Communications and Information Technologies (ISCIT)*, 2022 21st International Symposium on Communications and Information Technologies (ISCIT), Sept. 2022, pp. 1–6, DOI: 10.1109/ISCIT55906.2022.9931222, URL: https://ieeexplore.ieee.org/abstract/document/9931222 (visited on 04/12/2024).

[TKM20] Mineto Tsukada, Masaaki Kondo, and Hiroki Matsutani, « A Neural Network-Based On-device Learning Anomaly Detector for Edge Devices », *in*: *IEEE Transactions on Computers* (2020), pp. 1–1, ISSN: 0018-9340, 1557-9956, 2326-3814, DOI: 10.1109/TC.2020.2973631, URL: https://ieeexplore.ieee.org/document/9000710/ (visited on 10/23/2021).

[Tol+20] Vale Tolpegin *et al.*, « Data Poisoning Attacks Against Federated Learning Systems », *in*: *Computer Security – ESORICS 2020*, ed. by Liqun Chen *et al.*, Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 480–501, ISBN: 978-3-030-58951-6, DOI: 10.1007/978-3-030-58951-6_24.

[TR18] Wiem Tounsi and Helmi Rais, « A Survey on Technical Threat Intelligence in the Age of Sophisticated Cyber Attacks », *in*: *Computers & Security* 72 (Jan. 2018), pp. 212–233, ISSN: 01674048, DOI: 10.1016/j.cose.2017.09.001, URL: https://doi.org/10.1016/j.cose.2017.09.001.

[Uet+21] Rafael Uetz *et al.*, « Reproducible and Adaptable Log Data Generation for Sound Cybersecurity Experiments », *in*: *Annual Computer Security Applications Conference*, ACSAC '21: Annual Computer Security Applications Conference, Virtual Event USA: ACM, Dec. 6, 2021, pp. 690–705, ISBN: 978-1-4503-8579-4, DOI: 10.1145/3485832.3488020, URL: https://dl.acm.org/doi/10.1145/3485832.3488020 (visited on 08/08/2022).

[VKF15] Emmanouil Vasilomanolakis, Shankar Karuppayah, and Mathias Fischer, « Taxonomy and Survey of Collaborative Intrusion Detection », *in*: *ACM Computing Surveys* 47.*4* (May 2015), p. 33, DOI: 10.1145/2716260.

[Vy+21] Nguyen Chi Vy *et al.*, « Federated Learning-Based Intrusion Detection in the Context of IIoT Networks: Poisoning Attack and Defense », *in*: *Network and System Security*, ed. by Min Yang, Chao Chen, and Yang Liu, vol. 13041, Cham: Springer International Publishing, 2021, pp. 131–147, ISBN: 978-3-030-92707-3 978-3-030-92708-0, DOI: 10.1007/978-3-030-92708-0_8, URL: https://link.springer.com/10.1007/978-3-030-92708-0_8 (visited on 03/05/2024).

[Wag+19] Thomas D. Wagner *et al.*, « Cyber Threat Intelligence Sharing: Survey and Research Directions », *in*: *Computers & Security* 87 (2019), p. 101589, ISSN: 01674048, DOI: 10.1016/j.cose.2019.101589.

[Wag19] Thomas D Wagner, « Cyber Threat Intelligence for "Things" », *in*: *2019 International Conference on Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, IEEE, June 2019, pp. 1–2, ISBN: 978-1-72810-232-0, DOI: 10.1109/CyberSA.2019.8899384, URL: https://ieeexplore.ieee.org/document/8899384/.

[Wan+22] Ning Wang, Yang Xiao, *et al.*, « FLARE: Defending Federated Learning against Model Poisoning Attacks via Latent Space Representations », *in*: *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki Japan: ACM, May 30, 2022, pp. 946–958, ISBN: 978-1-4503-9140-5, DOI: 10.1145/3488932.3517395, URL: https://dl.acm.org/doi/10.1145/3488932.3517395 (visited on 07/05/2022).

[Web12] J. Beau W. Webber, « A Bi-Symmetric Log Transformation for Wide-Range Data », *in*: *Measurement Science and Technology* 24.*2* (Dec. 2012), p. 027001, ISSN: 0957-0233, DOI: 10.1088/0957-0233/24/2/027001, URL: https://dx.doi.org/10.1088/0957-0233/24/2/027001 (visited on 03/13/2024).

[WK21] Yuwei Wang and Burak Kantarci, « Reputation-Enabled Federated Learning Model Aggregation in Mobile Platforms », *in*: *ICC 2021 - IEEE International Conference on Communications*, ICC 2021 - IEEE International Conference on Communications, June 2021, pp. 1–6, DOI: 10.1109/ICC42927.2021.9500928.

[XL04] Li Xiong and Ling Liu, « PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities », *in*: *IEEE Transactions on Knowledge and Data Engineering* 16.7 (July 2004), pp. 843–857, ISSN: 1558-2191, DOI: 10.1109/TKDE.2004.1318566, URL: https://ieeexplore.ieee.org/document/1318566 (visited on 01/08/2024).

[XTL21] Qi Xia, Zeyi Tao, and Qun Li, « ToFi: An Algorithm to Defend Against Byzantine Attacks in Federated Learning », *in*: *Security and Privacy in Communication Networks*, ed. by Joaquin Garcia-Alfaro *et al.*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Cham: Springer International Publishing, 2021, pp. 229–248, ISBN: 978-3-030-90019-9, DOI: 10.1007/978-3-030-90019-9_12.

[Yad19] Omry Yadan, *Hydra - A Framework for Elegantly Configuring Complex Applications*, Github, 2019, URL: https://github.com/facebookresearch/hydra.

[Yan+19] Qiang Yang, Yang Liu, *et al.*, « Federated Machine Learning: Concept and Applications », *in*: *ACM Transactions on Intelligent Systems and Technology* 10.2 (Feb. 28, 2019), pp. 1–19, ISSN: 2157-6904, DOI: 10.1145/3298981, URL: https://dl.acm.org/doi/10.1145/3298981.

[Yan+23] Run Yang, Hui He, *et al.*, « Dependable Federated Learning for IoT Intrusion Detection against Poisoning Attacks », *in*: *Computers & Security* 132 (Sept. 1, 2023), p. 103381, ISSN: 0167-4048, DOI: 10.1016/j.cose.2023.103381, URL: https://www.sciencedirect.com/science/article/pii/S0167404823002912 (visited on 03/04/2024).

[Ye+23] Chuyao Ye *et al.*, « PFedSA: Personalized Federated Multi-Task Learning via Similarity Awareness », *in*: *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS), May 2023, pp. 480–488, DOI: 10.1109/IPDPS54959.2023.00055, URL: https://ieeexplore.ieee.org/document/10177489 (visited on 12/05/2023).

[Yin+18] Dong Yin *et al.*, « Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates », *in*: *Proceedings of the 35th International Conference on Machine Learning*, International Conference on Machine Learning, PMLR, July 3, 2018, pp. 5650–5659, URL: https://proceedings.mlr.press/v80/yin18a.html (visited on 03/07/2023).

[You+22] XinTong You *et al.*, « Poisoning Attack Detection Using Client Historical Similarity in Non-Iid Environments », *in*: *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Jan. 2022, pp. 439–447, DOI: 10.1109/Confluence52989.2022.9734158.

[Zha+19] Ying Zhao, Junjun Chen, *et al.*, « Multi-Task Network Anomaly Detection Using Federated Learning », *in*: *Proceedings of the Tenth International Symposium on Information and Communication Technology - SoICT 2019*, The Tenth International Symposium, Hanoi, Ha Long Bay, Viet Nam: ACM Press, 2019, pp. 273–279, ISBN: 978-1-4503-7245-9, DOI: 10.1145/3368926.3369705, URL: http://dl.acm.org/citation.cfm?doid=3368926.3369705 (visited on 06/07/2021).

[Zha+20a] Weishan Zhang, Qinghua Lu, *et al.*, « Blockchain-Based Federated Learning for Device Failure Detection in Industrial IoT », *in*: *IEEE Internet of Things Journal* (Sept. 6, 2020), pp. 1–1, ISSN: 2327-4662, DOI: 10.1109/JIOT.2020.3032544, URL: https://ieeexplore.ieee.org/document/9233457/.

[Zha+20b] Weishan Zhang, Tao Zhou, *et al.*, « Dynamic Fusion Based Federated Learning for COVID-19 Detection », *in*: *arXiv* 14.*8* (Sept. 22, 2020), pp. 1–9, ISSN: 23318422, URL: http://arxiv.org/abs/2009.10401.

[Zha+20c] Lingchen Zhao, Shengshan Hu, *et al.*, *Shielding Collaborative Learning: Mitigating Poisoning Attacks through Client-Side Detection*, Mar. 9, 2020, arXiv: 1910.13111 [cs], URL: http://arxiv.org/abs/1910.13111 (visited on 08/28/2022), prepublished.

[Zha+22a] Yuemeng Zhang, Yong Zhang, Zhao Zhang, *et al.*, « Evaluation of Data Poisoning Attacks on Federated Learning-Based Network Intrusion Detection System », *in*: *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), Dec. 2022, pp. 2235–2242, DOI: 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00330, URL: https://ieeexplore.ieee.org/document/10074658 (visited on 10/31/2023).

[Zha+22b] Zhao Zhang, Yong Zhang, Da Guo, *et al.*, « SecFedNIDS: Robust Defense for Poisoning Attack against Federated Learning-Based Network Intrusion Detection System », *in*: *Future Generation Computer Systems* 134 (Sept. 2022), pp. 154–169, ISSN: 0167739X, DOI: 10.1016/j.future.2022.04.010, URL: https://linkinghub.elsevier.com/retrieve/pii/S0167739X22001339 (visited on 07/05/2022).

[Zho+22] Jun Zhou, Nan Wu, *et al.*, « A Differentially Private Federated Learning Model against Poisoning Attacks in Edge Computing », *in*: *IEEE Transactions on Dependable and Secure Computing* (2022), pp. 1–1, ISSN: 1941-0018, DOI: 10.1109/TDSC.2022.3168556.

[ZLK10] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera, « A Survey of Coordinated Attacks and Collaborative Intrusion Detection », *in*: *Computers & Security* 29.*1* (Feb. 2010), pp. 124–140, ISSN: 01674048, DOI: 10.1016/j.cose.2009.06.008, URL: https://linkinghub.elsevier.com/retrieve/pii/S016740480900073X (visited on 07/21/2021).

[ZY15] Fatima Zohra Filali and Belabbes Yagoubi, « Global Trust: A Trust Model for Cloud Service Selection », *in*: *International Journal of Computer Network and Information Security* 7.*5* (Apr. 8, 2015), pp. 41–50, ISSN: 20749090, 20749104, DOI: 10.5815/ijcnis.2015.05.06, URL: http://www.mecs-press.org/ijcnis/ijcnis-v7-n5/v7n5-6.html (visited on 07/03/2024).

List of Figures

List of Tables

List of Acronyms

AASR Absolute Attack Success Rate. 84, 92, 93

AE Autoencoder. 15, 16

AI Artificial Intelligence. 3

ANN Artificial Neural Network. 12

ARES International Conference on Availability, Reliability and Security. 78

ASR Attack Success Rate. 84, 91, 92, 100, 117, 119, 120, 121, 122, 123, 172, 176

AV Autonomous Vehicles. 48, 50

BIRCH Balanced Iterative Reducing and Clustering using Hierarchies. 55, 57

BNN Binarized Neural Network. 55

BYOD Bring Your Own Device. 130

CD-FL Cross-Device Federated Learning. 26, 27

CERT Computer Emergency Response Team. 54

CIDN Collaborative Intrusion Detection Network. 20

CIDS Collaborative Intrusion Detection System. 4, 5, 11, 20, 22, 23, 28, 29, 30, 31, 37, 43, 51, 54, 58, 65, 66, 67, 69, 71, 73, 77, 78, 97, 104, 105, 115, 171

CNN Convolutional Neural Network. 14, 51

CPS Cyber-Physical System. 48

CS-FL Cross-Silo Federated Learning. 26, 65, 66, 103, 104

CSP Constraint Satisfaction Problem. 132, 133, 135

DAE Denoising Autoencoder. 16

DBN Deep Belief Network. 16

DDoS Distributed Denial of Service. 37

DL Deep Learning. 3, 12, 13, 15, 16, 18, 49, 61, 171

DLT Distributed Ledger Technology. 53

DNN Deep Neural Network. 13, 51, 110

DoS Denial of Service. 51, 56, 129

DP Differential Privacy. 61

DPR Data Poisoning Rate. 81, 82, 94, 95, 100, 106

DR Detection Rate. 70, 72

DT Decision Tree. 37

FIDS Federated Intrusion Detection System. 4, 6, 32, 33, 34, 37, 38, 39, 40, 41, 42, 43, 44, 45, 48, 50, 51, 53, 55, 56, 58, 59, 60, 61, 62, 63, 66, 73, 77, 83, 101, 103, 126, 127, 128, 129, 131, 140, 171, 173, 181

FL Federated Learning. 3, 4, 5, 6, 11, 20, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 34, 36, 37, 38, 39, 40, 43, 45, 48, 51, 53, 54, 55, 57, 59, 60, 61, 62, 65, 67, 68, 69, 70, 71, 72, 73, 77, 78, 79, 81, 95, 96, 99, 100, 104, 105, 106, 107, 108, 109, 110, 112, 115, 119, 124, 125, 171

FML Federated Mimic Learning. 54

FN False Negative. 19

FNR False Negative Rate. 19, 81

FP False Positive. 19, 84

FPR False Positives Rate. 19, 81

FTL Federated Transfer Learning. 27, 54, 56, 60

GAN Generative Adversarial Networks. 60, 100

GD Gradient Descent. 53, 54

HFL Horizontal Federated Learning. 27, 38, 39, 54

HIDS Host-based Intrusion Detection System. 48

IAT Inter-Arrival Time. 17, 49

ICS Industrial Control System. 4, 39, 41, 42

IDS Intrusion Detection System. 3, 4, 5, 6, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23, 30, 31, 32, 34, 36, 37, 38, 40, 56, 57, 59, 62, 65, 66, 68, 71, 77, 79, 81, 92, 94, 99, 100, 105

IID Independent and Identically Distributed. 28, 37, 48, 49, 57, 63, 68, 69, 81, 82, 83, 95, 100, 107, 124

IIoT Industrial Internet of Things. 38

IoMT Internet of Medical Things. 42

IoT Internet of Things. 4, 18, 26, 37, 38, 39, 41, 42, 48, 49, 50, 57, 91, 130

ISAC Information Sharing and Analysis Center. 4

IT Information Technology. 4, 44, 48

KLD Kullback-Leibler Divergence. 108

LDA Latent Dirichlet Allocation. 35

MAPE-K Monitor-Analyze-Plan-Execute plus Knowledge. 44

MCC Mathew Correlation Coefficient. 20

MEC Mobile Edge-Computing. 50

MitM Man-in-the-Middle. 133

ML Machine Learning. 3, 11, 12, 13, 14, 15, 16, 17, 18, 20, 22, 23, 24, 30, 31, 33, 36, 37, 43, 44, 49, 50, 51, 53, 58, 59, 61, 62, 99, 100, 105, 111

MLP Multilayer Perceptron. 14, 51, 80, 116, 171

MPC Multi-Party Computation. 38, 61

MPR Model Poisoning Rate. 82, 94, 95, 100, 106

MSE Mean Squared Error. 16

MTA Main Task Accuracy. 84, 92, 94

MTL Multi-Task Learning. 54, 56

MUD Manufacturer Usage Description. 55

NIDS Network-based Intrusion Detection System. 4, 12, 13, 17, 69, 79, 106, 115, 124, 128, 175

NIID Non Independent and Identically Distributed. 28, 57, 66, 68, 69, 70, 83, 105, 107, 108, 109, 116, 118, 119, 123, 127

NMF Non-negative Matrix Factorization. 35, 173, 181

NN Neural Network. 51

P2P Peer-to-Peer. 66

PCA Principal Component Analysis. 49, 79, 80, 95, 97, 98, 99, 171, 172

PPV Positive Predictive Value. 19

RASR Relative Attack Success Rate. 84, 85, 92, 93, 94, 95, 96, 172

RBM Restricted Boltzmann Machine. 16

RF Random Forest. 12, 37

RNN Recurrent Neural Network. 14, 51

RQ Research Question. 33

RSU Roadside Unit. 53

SAE Stacked Autoencoder. 15, 16, 171

SDN Software-Defined Networking. 42, 50, 51, 61, 131

SGD Stochastic Gradient Descent. 13, 24, 25

SLR Systematic Literature Review. 5, 6, 31, 32, 34, 38, 59

SOC Security Operations Center. 4, 22

SPoF Single Point-of-Failure. 22, 53

SRDS IEEE International Symposium on Reliable Distributed Systems. 104

SVM Support Vector Machine. 12, 37

TI Threat Intelligence. 22

TL Transfer Learning. 56

TN True Negative. 19, 84

TNR True Negative Rate. 19

TNSM IEEE Transactions on Network and Service Management. 32

TP True Positive. 19, 84

TPR True Positives Rate. 19

TSP Topology Selection Problem. 133

V2X Vehicle-to-Everything. 42, 53

VFL Vertical Federated Learning. 27, 54

VPN Virtual Private Network. 56, 130

APPENDICES

A Additional figures

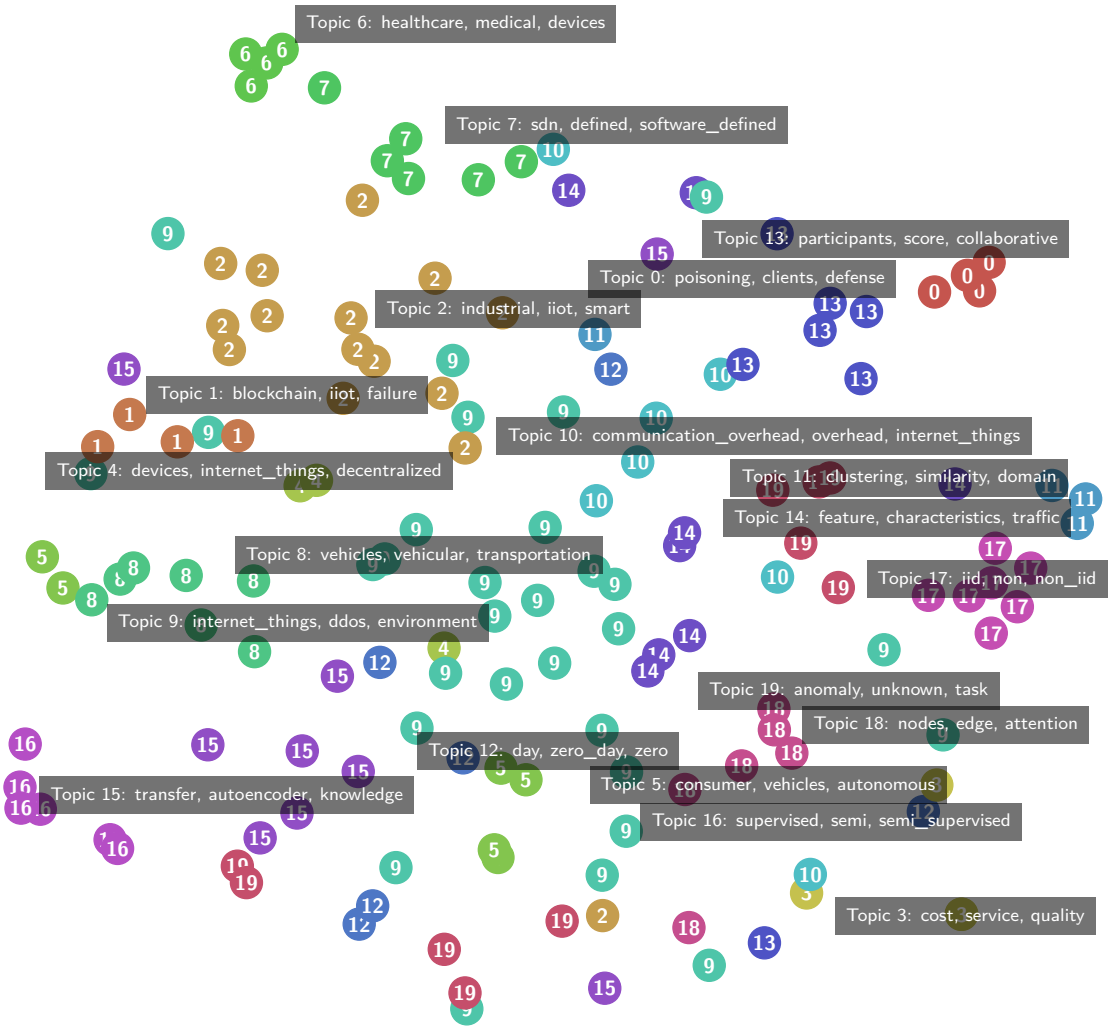


Figure 8.1 – Topic embedding of the Federated Intrusion Detection System (FIDS) literature using a Non-negative Matrix Factorization (NMF) model with 20 topics. Each point represents a paper, and each are labelled with the topic they are the most associated with.

B Résumé en français de la thèse

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Améliorer la détection d'intrusions dans les systèmes répartis grâce à l'apprentissage fédéré

Mot clés : apprentissage fédéré, détection d'intrusions collaborative, systèmes répartis, hétérogénéité des données, confiance et réputation

Résumé : La collaboration entre les différents acteurs de la cybersécurité est essentielle pour lutter contre des attaques de plus en plus sophistiquées et nombreuses. Pourtant, les organisations sont souvent réticentes à partager leurs données, par peur de compromettre leur confidentialité et leur avantage concurrentiel, et ce même si cela pourrait d'améliorer leurs modèles de détection d'intrusions. L'apprentissage fédéré est un paradigme récent en apprentissage automatique qui permet à des clients répartis d'entraîner un modèle commun sans partager leurs données. Ces propriétés de collaboration et de confidentialité en font un candidat idéal pour des applications sensibles comme la détection d'intrusions. Si un certain nombre d'applications ont montré qu'il est, en effet, possible d'entraîner un modèle unique sur des données réparties de détection d'intrusions, peu se sont intéressées à l'aspect collaboratif de ce paradigme. Dans ce manuscrit, nous étudions l'utilisation de l'apprentissage fédéré pour construire des systèmes collaboratifs de détection d'intrusions. En particulier, nous explorons (i) l'impact de la qualité des données dans des contextes hétérogènes, (ii) l'exposition à certains types d'attaques par empoisonnement, et (iii) des outils et des méthodologies pour améliorer l'évaluation de ce type d'algorithmes.

Title: Improving Intrusion Detection in Distributed Systems with Federated Learning

Keywords: federated learning, collaborative intrusion detection, distributed systems, data heterogeneity, trust and reputation

Abstract: Collaboration between different cybersecurity actors is essential to fight against increasingly sophisticated and numerous attacks. However, stakeholders are often reluctant to share their data, fearing confidentiality and privacy issues and the loss of their competitive advantage, although it would improve their intrusion detection models. Federated learning is a recent paradigm in machine learning that allows distributed clients to train a common model without sharing their data. These properties of collaboration and confidentiality make it an ideal candidate for sensitive applications such as intrusion detection. While several applications have shown that it is indeed possible to train a single model on distributed intrusion detection data, few have focused on the collaborative aspect of this paradigm. In this manuscript, we study the use of federated learning to build collaborative intrusion detection systems. In particular, we explore (i) the impact of data quality in heterogeneous contexts, (ii) the exposure to certain types of poisoning attacks, and (iii) tools and methodologies to improve the evaluation of these types of algorithms.