COLLEGES BRETAGNE LOIRE | SCIENCES POUR L'INGENIEUR 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 des systèmes distribués grâce à l'apprentissage fédéré

Thèse présentée et soutenue à Rennes, le XX septembre 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 CNRS au laboratoire LIRIS
	Pierre-François Gimenez	Maître de Conférence à CentraleSupelec
	Vincent Nicomette	Professeur à l'INSA de Toulouse
	Fabien Autrel	Ingénieur de Recherche à IMT Atlantique
	Marc-Oliver Pahl	Directeur d'Études à IMT Atlantique
Dir. de thèse :	Yann Busnel	Directeur de la Recherche et de l'Innovation (DRI) 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 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 distribués 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 distribuées de détection d'intrusions, peu se sont intéressées à l'aspect collaboratif de ce paradigme. En plus de l'aspect collaboratif, d'autres problématiques apparaissent dans ce contexte, telles que l'hétérogénéité des données des différents participants ou la gestion de participants non fiables. Dans ce manuscrit, nous explorons l'utilisation de l'apprentissage fédéré pour construire des systèmes collaboratifs de détection d'intrusions. En particulier, nous explorons l'impact de la qualité des données dans des contextes hétérogènes, certains types d'attaques par empoisonnement, et proposons des outils et des méthodologies pour améliorer l'évaluation de ce type d'algorithmes distribués.

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, 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 addition to the collaborative aspect, other challenges arise in this context, such as the heterogeneity of the data between different participants or the management of untrusted contributions. In this manuscript, we explore the use of federated learning to build collaborative intrusion detection systems. In particular, we explore the impact of data quality in heterogeneous contexts, some types of poisoning attacks, and propose tools and methodologies to improve the evaluation of these types of distributed algorithms.

ACKNOWLEDGEMENTS

Table of Contents

Part I

Federated Learning to build CIDSs

Chapter 1

Background and Preliminaries

Contents

1.1 Introduction

This chapter provides the necessary background and preliminaries to understand the rest of the thesis. We start by laying out the basics of Machine Learning (ML) for intrusion detection in Section 1.2, followed by the implications of scaling up to Collaborative Intrusion Detection Systems (CIDSs) in Section 1.3, enumerating along the way the challenges that motivate the use of Federated Learning (FL). We then introduce the fundamentals of FL in Section 1.4, focusing on the FedAvg algorithm and the notations used throughout the thesis. Finally, we discuss the threats against FL in **??**, with a particular focus on data poisoning attacks.

1.2 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 1.1 displays an example of a signature for detecting the Heartbleed vulnerability using Suricata, 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. 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-based IDSs fall into this category. On the other hand, anomaly detection compares a

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

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.

1.2.1 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 and destination addresses, ports, and protocol. Various features can be extracted from

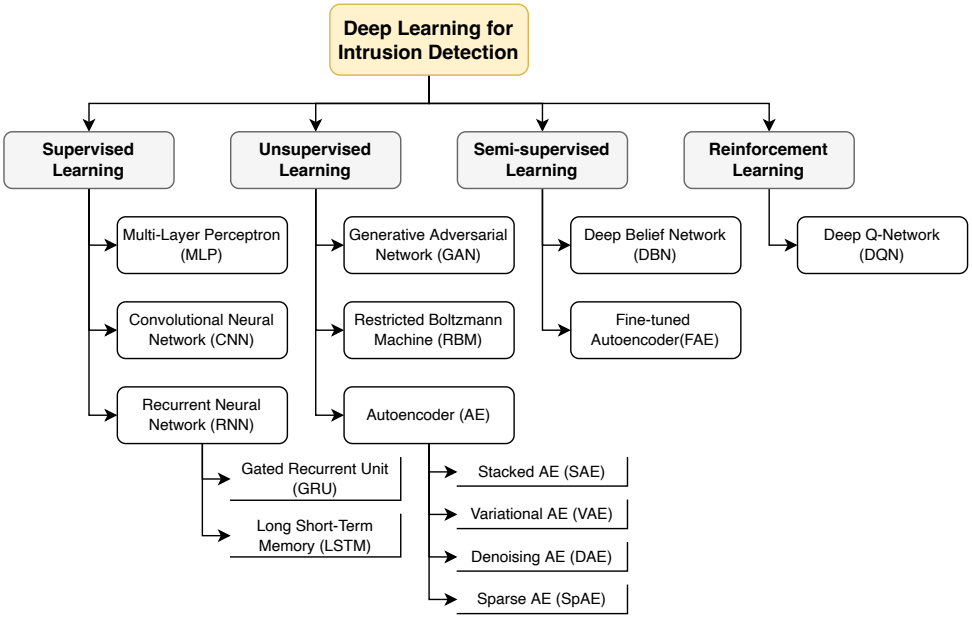


Figure 1.1 – Taxonomy of the main DL paradigms.

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 1.2.2. More generally, the dataset (D) is represented as a set of variables $X_i = \langle x_1, x_2, \ldots, x_n \rangle, i \in [\![1, n]\!]$, where x_j corresponds to the j-th feature.

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 1.1 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 [**?**], 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 loss function

$$\mathcal{L}(w, X_i, y_i), i \in [\![1, n]\!]. \tag{1.1}$$

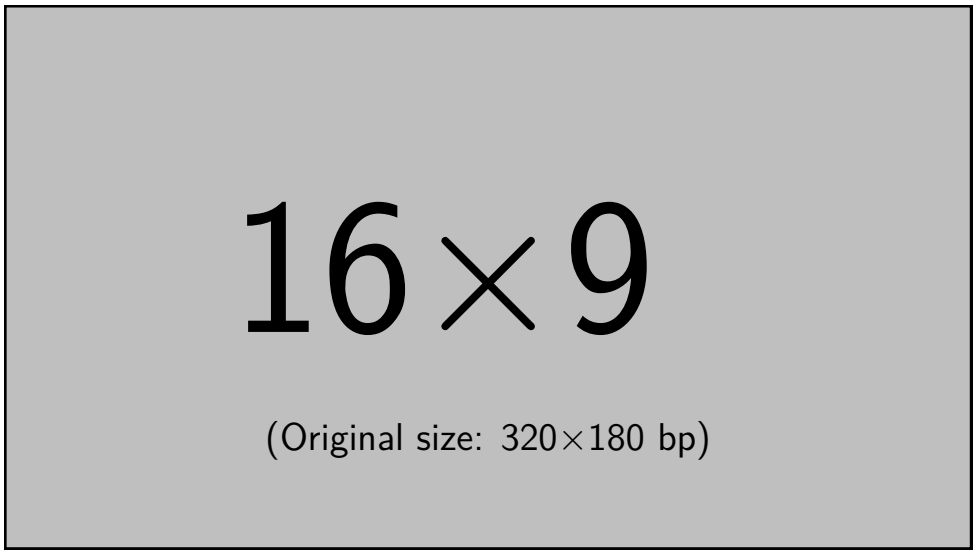


Figure 1.2 – Workflow of training a Multilayer Perceptron (MLP) for intrusion detection.

After computing the gradients $\nabla\mathcal{L}(w, X_i, y_i)$, they can update their model as

$$w^{t+1} \leftarrow w^t - \eta\nabla\mathcal{L}(w, X_i, y_i), \tag{1.2}$$

where η is the learning rate, and w the model's parameters, and t the iteration. 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.

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 models that are skewed by the class distribution.

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.

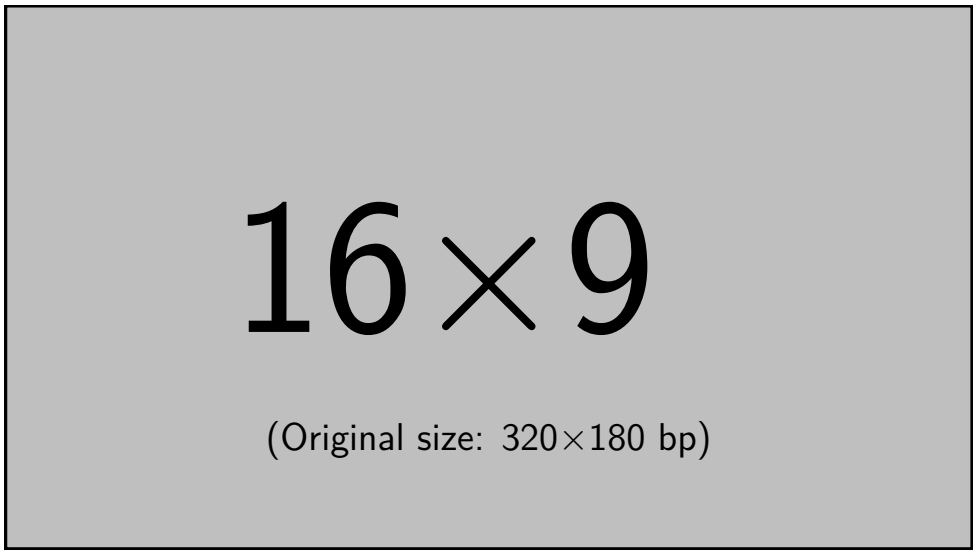


Figure 1.3 – Workflow of a Stacked Autoencoder (SAE) for intrusion detection.

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 $X = \{X_i | i \in [\![1, n]\!]\}$, the model is trained to minimize the reconstruction error $\mathcal{L}(X, \hat{X})$, where $\hat{X}$ is the output of the AE. A typical error function for this task is the Mean Squared Error (MSE), expressed as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(X_i - \hat{X}_i \right)^2 . \tag{1.3}$$

Then the model can be updated using the same process described in Equation (1.2). 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. To detect anomalies, the reconstruction error of a new sample is compared to a threshold τ 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 model and its 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 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.

1.2.2 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 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 1.1 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) [MS15; Kor+19; Mou21] 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.

3. 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.

Table 1.1 – 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.

Provided features Because most of the datasets presented in Table 1.1 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 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 relates to the selection of the relevant features for a given task. 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 based on 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 NF-V2 datasets 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.

4. Formerly known as Bro, available at: https://www.zeek.org/
5. Available at: https://openargus.org/argus-ids
6. Available at: https://www.ntop.org/products/traffic-analysis/nprobe/

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True Positives (TPs)	False Negatives (FNs)
	Negative	False Positives (FPs)	True Negatives (TNs)

Table 1.2 – Confusion matrix for binary classification.

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].

1.2.3 Metrics

Most research on ML for intrusion detection relies on the same set of metrics to assess, validate, and compare their solutions [Cha+19; Far+20; BG16]. Most of these metrics are derived from the confusion matrix (see Table 1.2), 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.

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

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

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

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

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

(4) *Specificity*, or True Negative Rate (TNR), is the proportion of negative cases that

has been correctly categorized.

$$Specificity = \frac{TN}{P} = \frac{TN}{TN + FP}$$

(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.

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

(6) *Miss rate*, 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.

$$Miss\ rate = \frac{FN}{P} = \frac{FN}{FN + TP}$$

(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].

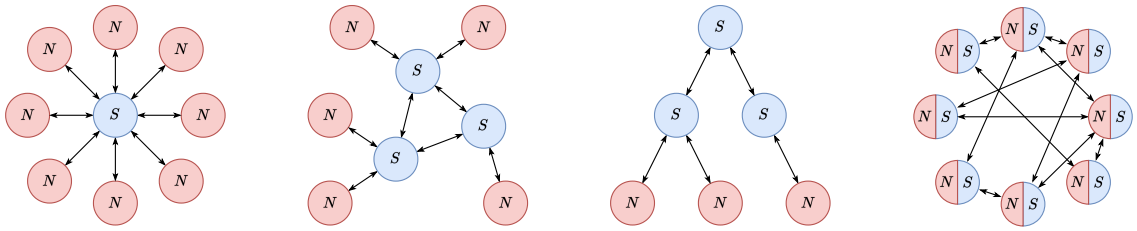
$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

(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.

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

1.3 Collaboration in Intrusion Detection

The topic of collaboration in intrusion detection is rather old, with several surveys and reviews available in the literature [ZLK10; EO11; VKF15; Men+15; FS16; LMK22], 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, also 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.



(a) Centralized **(b)** Decentralized **(c)** Hierarchical **(d)** Distributed

Figure 1.4 – 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.

This distinction also transpires from 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 1.4).

1.3.1 The different topologies

The aforementioned literature identify three main types of topologies for CIDSs: centralized, decentralized, and distributed. 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 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 1.4. 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:

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

Decentralized In a decentralized architecture, multiple servers coexist. Each server is responsible for a subset of the nodes, and they can share information between them.

Hierarchical A hierarchical architecture is decentralized system 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.

Distributed In a distributed 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 1.4 illustrates these topologies. In the centralized architecture (Figure 1.4a), all nodes are connected to a single server. Figure 1.4b shows a standard example of a decentralized architecture. Figures 1.4c and 1.4d illustrate the specific cases of decentralized architectures: hierarchical and distributed, respectively. 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.

1.3.2 Challenges in Collaborative Intrusion Detection

Collaborative intrusion detection as a whole faces several challenges, some of which vary depending on the topology. The first intuitive topology-related challenge is the Single Point-of-Failure (SPoF) that a centralized architecture represents [**<empty citation>**]. If the analysis is performed remotely, like in a Security Operations Center (SOC) monitoring consumers' infrastructures, a failure on the central server would (at least partially) prevent detection. Fortunately, the use case of knowledge sharing implies that (some of) the detection is performed locally and the central server is only used for updates and coordination. This reduces the impact of a centralized failure, but still implies that the collaboration is conditioned by the availability of the central server.

Collaborative intrusion detection faces challenges, including the SPoF in a centralized architectures. If the analysis is performed remotely, a failure on the central server would hinder detection. However, 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. Typical CIDSs are centralized, and therefore 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 Agency for Cybersecurity, defines the actionability of Threat Intelligence (TI) as the fulfillment of five criteria: relevance, digestibility, accuracy, completeness, and timeliness [ENI14]. The latter is directly provided by the supporting architecture. 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 increase 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. This is especially critical, as without appropriate measures, this

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), as well as reputation loss [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].

1.4 Fundamentals of Federated Learning

1.4.1 Types of Federated Learning

1.4.2 The `FedAvg` Algorithm

1.4.3 The Question of Data Distribution

1.4.4 Threats against Federated Learning

1.5 Conclusion

Part II

Quantifying the Limitations of FIDSs

Part III

Providing Solutions

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).

[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).

[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.

[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).

[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).

[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/.

[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).

[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.

[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).

[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).

[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).

[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.

[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).

[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).

[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).

[Men+15] Guozhu Meng *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).

[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.

[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).

[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.

[Rin+17a] Markus Ring *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 *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.

[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).

[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 Associa-

tion, June 1992, URL: https://www.usenix.org/conference/usenix-summer-1992-technical-conference/dids-distributed-intrusion-detection-system.

[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/.

[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.

[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).

List of Figures

List of Tables

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 des systèmes distribués grâce à l'apprentissage fédéré

Mot clés : apprentissage automatique, apprentissage fédéré, détection d'intrusions, collaboration, confiance

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 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 distribués 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 distribuées de détection d'intrusions, peu se sont intéressées à l'aspect collaboratif de ce paradigme. En plus de l'aspect collaboratif, d'autres problématiques apparaissent dans ce contexte, telles que l'hétérogénéité des données des différents participants ou la gestion de participants non fiables. Dans ce manuscrit, nous explorons l'utilisation de l'apprentissage fédéré pour construire des systèmes collaboratifs de détection d'intrusions. En particulier, nous explorons l'impact de la qualité des données dans des contextes hétérogènes, certains types d'attaques par empoisonnement, et proposons des outils et des méthodologies pour améliorer l'évaluation de ce type d'algorithmes distribués.

Title: Improving Intrusion Detection in Distributed Systems with Federated Learning

Keywords: machine learning, federated learning, intrusion detection, collaboration, trust

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, 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 addition to the collaborative aspect, other challenges arise in this context, such as the heterogeneity of the data between different participants or the management of untrusted contributions. In this manuscript, we explore the use of federated learning to build collaborative intrusion detection systems. In particular, we explore the impact of data quality in heterogeneous contexts, some types

of poisoning attacks, and propose tools and methodologies to improve the evaluation of these types of distributed algorithms.