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. 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 (i) l'impact de la qualité des données dans des contextes hétérogènes, (ii) certains types d'attaques par empoisonnement, et (iii) proposons des outils et des méthodologies pour améliorer l'évaluation de ce type d'algorithmes répartis.

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 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 (i) the impact of data quality in heterogeneous contexts, (ii) some types of poisoning attacks, and (iii) 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

Part II

Quantifying the Limitations of FIDSs

Part III

Providing Solutions

Chapter 1

Model Quality Assessment for Reputation-aware Collaborative Federated Learning

Contents

1.1 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. **??** introduces the reader to the problem of model quality assessment in CS-FL and the necessary background. Section 1.3 reviews the related work, before we dive in RADAR's architecture in Section 1.4. Section 1.5 and **??** present the experimental setup and results, and we discuss our findings in Section 1.7. Finally, **??** 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.

1.2 Problem Statement

In continuity with ????, we consider once more the use case introduced in ?? and the associated datasets. Specifically, we focus on a heterogeneous declination 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 ??, 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.

1.2.1 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.

1.2.2 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 **??**, 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.

1.2.3 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 1.1: 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?

1.3 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.

1.3.1 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 *et al.* [Zha+20] 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.

1.3.2 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.

1.3.3 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 1.1 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 1.1: 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 **??**) 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 1.2 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.

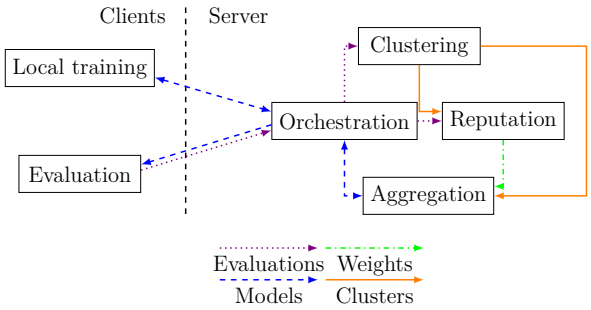


Figure 1.1 – *Architecture overview.*

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.

1.4 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 1.4.1), (*ii*) a similarity-based clustering algorithm that groups participants based on evaluations (Section 1.4.2), and (*iii*) a reputation system that assesses participants' trustworthiness based on their past contributions (Section 1.4.3). Figure 1.1 provides an overview of RADAR.

1.4.1 Assessing Contributions with Cross-Evaluation

As highlighted in Section 1.3, 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+20]. 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,

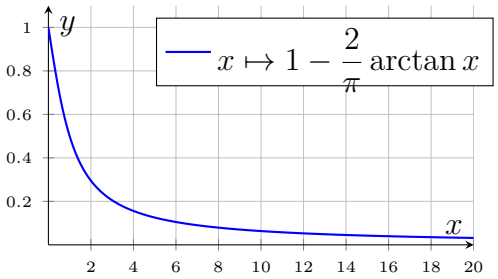


Figure 1.2 – Loss normalization function.

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 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* $E^r_{[i,*]}$ is used for clustering, while both the *received evaluations* vector $E^r_{[*,j]}$ and the *issued evaluations* vector $E^r_{[i,*]}$ are used in the reputation system. Algorithm 1.1 details the proposed workflow.

While most of the metrics for ML (see **??**) 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 1.2), as it emphasizes the lower part of the spectrum, where the differences between model losses are concentrated.

Algorithm 1.1 RADAR. R is the number of rounds, β the local batch size, η the learning rate, $\mathcal{E}$ 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 **??** in **??**).

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 1.4.2*
  for all $C_k^r \in \mathscr{C}^r$ **do**
    $(\rho_i^r) \leftarrow \textsc{ComputeReput}(E^r, \mathscr{C}^r)$ ▷ *See: Section 1.4.3*
    $\overline{W}^r \leftarrow \dfrac{1}{|C_k^r|} \displaystyle\sum_{i=0}^{|C_k^r|} w_i^r$

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

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

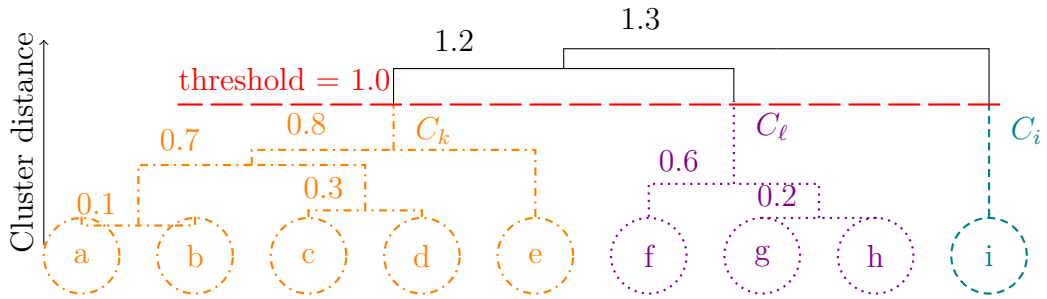


Figure 1.3 – Hierarchical clustering process.

1.4.2 Fighting Heterogeneity with Clustering

The clustering algorithm seeks to gather similar participants together in more homogeneous sub-federations when appropriate. Nguyen *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 $\delta^r_{i,j}$ the distance between the evaluations of p_i and p_j at round r. $\delta^r_{i,j}$ is defined 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 1.3 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:

$$\theta = \beta\overline{\Delta^r} = \frac{\beta}{|\mathscr{C}^r|(|\mathscr{C}^r|-1)} \sum_{\substack{k,\ell \in \mathscr{C}^r, \\ k \neq l}} \Delta^r_{k,\ell} \tag{1.1}$$

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 1.4.3. Con-

sequently, the server maintains as many global models $\overline{w}_k^r$ 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.

1.4.3 Ensuring Quality Contributions with Reputation

The reputation system centrally computes the weights $\rho_i^r, \forall p_i \in C_k^r$ used in the aggregation of each cluster model $\overline{w}_k^r$ at round r (see Section 1.4.1). 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 $E_{[*,i]}^r$ received by a participant p_i are continuous over $[0, 1]$, and thus need to be discretized into a set of q possible values $\mathcal{E} = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_q\}$.

A Dirichlet distribution on the outcome of an unknown event (*i.e.*, the mean of the received evaluation $\frac{1}{n} \sum_{e_{i,j}^r \in E_{[*,j]}^r} e_{i,j}^r$) 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 $\vec{\gamma}^r = \{\gamma_1^r, \gamma_2^r, \ldots, \gamma_q^r\}$ the cumulative evaluations received by p_i: $\gamma_2^r = 3$ means that three evaluations in $E_{[*,j]}^r$ had values bounded by $\left[\frac{1}{q}, \frac{2}{q}\right[$. We then note $\vec{\mathbb{P}} = \langle \mathbb{P}\{\varepsilon_1\}, \mathbb{P}\{\varepsilon_2\}, \ldots, \mathbb{P}\{\varepsilon_q\} \rangle$ the probability distribution vector for the received evaluation of a participant, where $\sum_{s=1}^{q} \mathbb{P}\{\varepsilon_s\} = 1$. Leveraging the cumulative evaluations $\vec{\gamma}^r$, the probability $\mathbb{P}\{\varepsilon_s | \vec{\gamma}^r\}$ is given by $\mathbb{P}\{\varepsilon_s | \vec{\gamma}^r\} = \gamma_s / \sum_{m=1}^{q} \gamma_m$.

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 $p_i \in C_k^r$ are weighted according to their similarity with other cluster members' [XL04] as $e'^r_{i,j} = e_{i,j}^r sim(E_{[i,*]}^r, E_{[C_k^r,*]}^r)$, where the similarity is defined as:

$$sim(E_{[i,*]}^r, E_{[C_k^r,*]}^r) = 1 - \sqrt{\frac{\sum_{j=1}^{n} \left(e_{i,j}^r - \sum_{i \in C_k^r} \frac{e_{i,j}^r}{|C_k^r|} \right)^2}{|P|}}. \tag{1.2}$$

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_i^r of a participant p_i at round r based on the prior knowledge γ_i^r of this participant is given by Equation (1.3). 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_i^r, the weight ρ_i^r of w_i^r for aggregation in $\overline{w}_k^r$ (see Step 3. in Section 1.4.1) is given by Equation (1.4).

$$\psi_i^r = \sum_{\kappa=1}^{r} \lambda^{r-\kappa} \gamma_i^\kappa \tag{1.3}$$

$$\rho_i^r = \frac{\psi_i^r}{\sum_j^{|C_k^r|} \psi_j^r} \tag{1.4}$$

As such, the weight ρ_i^r 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 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.

1.5 Experimental Setup

We evaluate RADAR and any selected baseline on a set of heterogeneous intrusion detection datasets [SLP22] with various attack scenarios (see Section 1.5.2). We implement the described use case (**??**) 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 1.1. 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.

1.5.1 Datasets and local algorithm

In continuity with the previous chapters, we implement our CIDS use case using the datasets introduced in **??** 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+21].

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

1. TODO.

Table 1.1 – *Hyperparameters.* The model's configuration is taken from the work of Popoola *et al.* [Pop+21], 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

Table 1.2 – *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			
		CIC-IDS	NB15	ToN_IoT	Bot-IoT
Training set	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**

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 *et al.* [Pop+21]. We reuse the hyperparameters provided by the authors (see Table 1.1), 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 1.2. 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.

1.5.2 Evaluation scenarios

The threat model defined in Section 1.2.3 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 1.5.1 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 (**??** 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.

1.5.3 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.

1.6 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 1.5.2. Like the others, RADAR is abbreviated as RA when needed.

1.6.1 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 dataset of origin. This partition is presented as Partition A in Table 1.3. 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

Table 1.3 – *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

Table 1.4 – *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

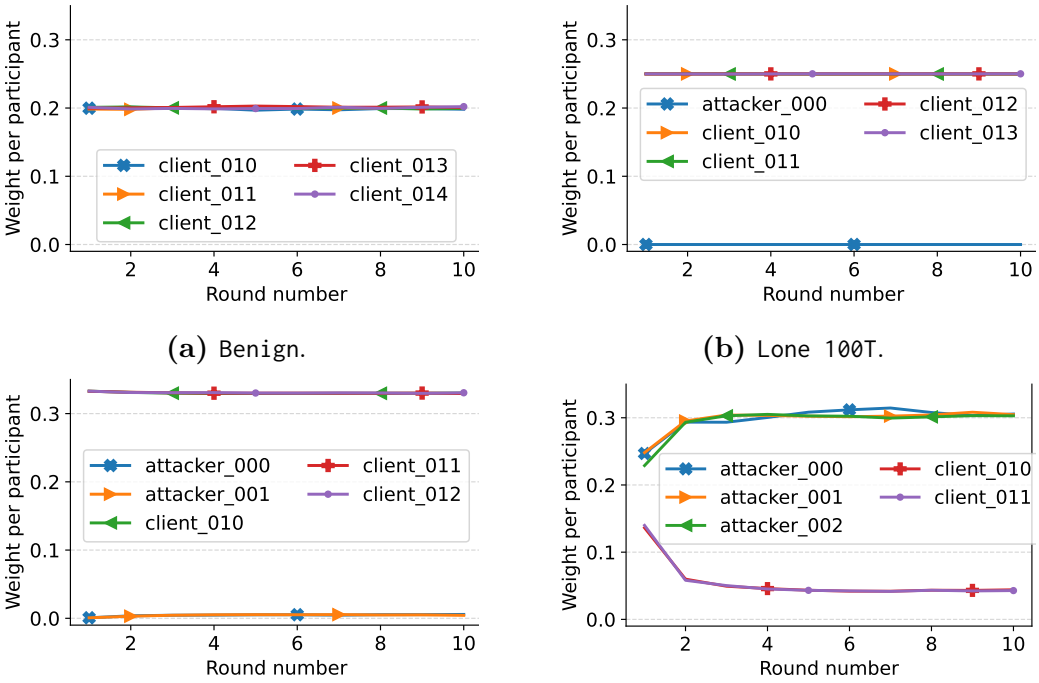
not unfairly penalize legitimate participants because of their potential differences. Figure 1.4a presents the weights provided by the reputation system for the aggregation. In the `Benign` scenario, the 5 participants originating from the Bot-IoT dataset do have equal weights, confirming that none of them is penalized by the reputation system. Furthermore, Table 1.4 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.

1.6.2 Handling data quality

Another goal for `RADAR` is to handle contributions of various quality. This objective is mostly represented by scenarios of **??** 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 1.3). 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 1.1).

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 1.4b, 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 1.4, where the ASR for both `Lone 100T` and `Lone 100U` are comparable to the benign case, underlining `RADAR` resilience. The results in Figure 1.5 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.



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

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

Figure 1.4 – *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.

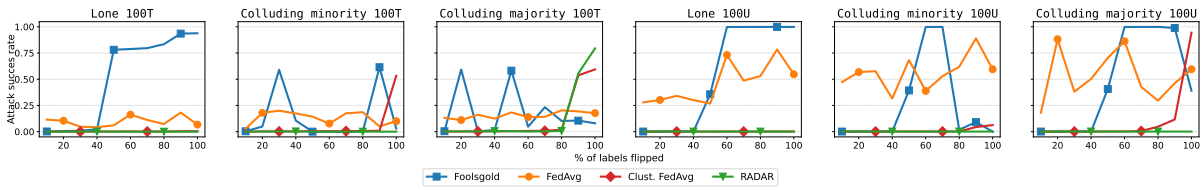


Figure 1.5 – *Attack Success Rate (ASR) of the different baselines.* Even though attackers are a majority, they gain weight precedence only for higher poisoning rates (>90%).

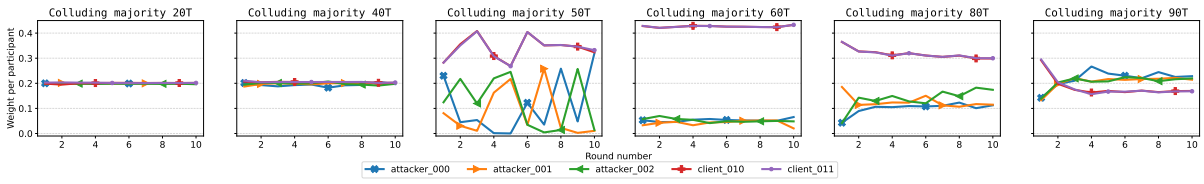


Figure 1.6 – *Aggregation weights ρ_i^r per participant of the poisoned cluster (*`Colluding majority T`*).* Even though attackers are a majority, they gain weight precedence only for higher poisoning rates (≥90%).

1.6.3 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 1.6.2, 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 1.3. 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.

Minority of attackers The `Colluding minority` class (**??** C3) contains scenarios where 2 out of 5 participants instantiated in Bot-IoT perpetrate label-flipping attacks. Here, the results depicted in Figure 1.4c indicate that the attackers are heavily penalized by the reputation system. This is coherent with the results in Table 1.4 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 1.4d 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 1.7: 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 1.5 and 1.6, 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 1.8 depicts how the reputation system reacts to participants that change their *noisiness* over time. Figure 1.8a 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 1.1). On the contrary, Figure 1.8b 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 1.5 and 1.6: 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.

1.6.4 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 1.5).

The results in Table 1.4 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.

1.7 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.

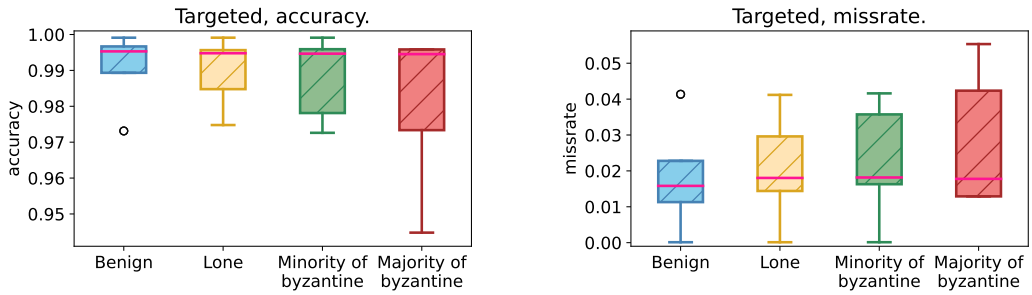
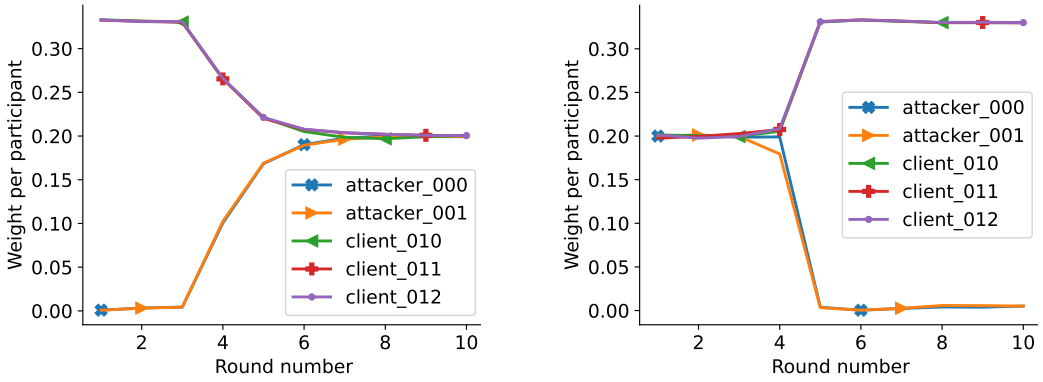


Figure 1.7 – RADAR*'s metric distribution among participants in different scenarios (*`100T`*).* The accuracy's and miss rate's lower bounds suddenly drop when attackers outnumber benign participants in the affected cluster. Indeed, clients in other clusters are unaffected by the poisoning.



(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 1.8 – *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.

1.7.1 Heterogeneity

The experiments conducted in **??** show that RADAR can handle heterogeneous participants. However, the simulation of the *practical* NIID setting is 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 **??** is expected to cope with this limitation, and thus represent an interesting research direction to further investigate heterogeneous settings.

1.7.2 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.

1.7.3 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 1.4.1) 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.

1.7.4 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.

1.7.5 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.

1.8 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.

BIBLIOGRAPHY

[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.

[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.

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

[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), prepublished.

[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+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.

[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).

[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).

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

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

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

[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.

[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).

[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).

[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.

[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).

[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).

[Léo+24] **Léo Lavaur** *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.

[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.

[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).

[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.

[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).

[Ngu+22] Thien Duc Nguyen *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).

[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).

[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.

[Pop+21] Segun I. Popoola *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.

[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).

[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).

[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).

[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.

[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.

[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).

[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.

[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).

[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+20] Lingchen Zhao *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), pre-published.

[Zho+22] Jun Zhou *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.

[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

COLLEGES BRETAGNE LOIRE SCIENCES POUR L'INGENIEUR 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 automatique, apprentissage fédéré, détection d'intrusions, collaboration, données hétérogènes, 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 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. 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 (i) l'impact de la qualité des données dans des contextes hétérogènes, (ii) certains types d'attaques par empoisonnement, et (iii) proposons des outils et des méthodologies pour améliorer l'évaluation de ce type d'algorithmes répartis.

Title: Improving Intrusion Detection in Distributed Systems with Federated Learning

Keywords: machine learning, federated learning, intrusion detection, collaboration, heterogeneous data, 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 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 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 (i) the impact of data quality in heterogeneous contexts, (ii) some types of poisoning attacks, and (iii) propose tools and methodologies to improve the evaluation of these types of distributed algorithms.