Apprentissage fédéré cherche modèles de qualité, bonne réputation exigée

Yann Busnel[1,2], Léo Lavaur[2,3,4,‡], Pierre-Marie Lechevalier[2,3,‡], Romaric Ludinard[2,3] et Géraldine Texier[2,3]

[1]*Institut Mines-Télécom, Palaiseau, France*
[2]*IRISA, Rennes, France*
[3]*IMT Atlantique, département SRCD, Cesson-Sévigné, France*
[4]*Interdisciplinary Centre for Security, Reliability and Trust (SnT), Université du Luxembourg*

L'apprentissage fédéré cross-silo (CS-FL) est un paradigme d'apprentissage collaboratif réparti qui permet à un groupe d'organisations d'entraîner un modèle commun. Cependant, ces systèmes sont vulnérables aux contributions de mauvaise qualité, rendant cruciales les protections contre les participants négligents ou malveillants. De plus, l'hétérogénéité des cas d'usage CS-FL complique la distinction entre contributions malveillantes et légitimes.

Pour répondre à ces défis, nous introduisons `RADAR`, une architecture de CS-FL capable d'évaluer la qualité des contributions, sans faire d'hypothèse sur la similarité des données. Grâce à une évaluation croisée, `RADAR` regroupe les participants perçus comme similaires et attribue un score de réputation pour pondérer leurs contributions lors de l'agrégation. Nous avons évalué notre approche dans des scénarios concrets de système collaboratif de détection d'intrusions (CIDS), en perturbant l'étiquetage des données pour simuler des variations de qualité. Les résultats montrent qu'en combinant partitionnement des participants et système de réputation, `RADAR` peut contrer un large éventail de comportements Byzantins, démontrant ainsi sa polyvalence.

Mots-clefs : Apprentissage fédéré, systèmes de réputation, empoisonnement de modèles, partitionnement, confiance.

1 Introduction

L'apprentissage collaboratif permet à plusieurs organisations d'entraîner un modèle unique à partir des informations détenues par chacune d'entre elles. Son déploiement est resté restreint, notamment à cause des risques induits par le partage de données. L'apprentissage fédéré (FL) répond à cette problématique en entraînant localement un modèle commun, permettant ainsi à ses participants de collaborer tout en gardant l'exclusivité de l'accès à ses données. Cette configuration, où les participants représentent un faible nombre de sources de données indépendantes, durables et hétérogènes, est souvent décrite comme apprentissage fédéré cross-silo (CS-FL). La construction collaborative de systèmes de détection d'intrusions (IDSs) en est un exemple pratique, les données du trafic réseau étant souvent considérées comme sensibles. Dans ce contexte, les données des organisations sont très influencées par les différences de cas d'usage ou de protocoles utilisés au sein de leurs infrastructures, au point de compliquer l'agrégation. Les systèmes collaboratifs sont également exposés à des contributions pouvant être de mauvaise qualité ou manipulées par des participants malveillants, ce qui accroît les risques. Identifier ces contributions est rendu particulièrement difficile par l'hétérogénéité élevée dans ce type de configuration : comment différencier une contribution malveillante ou erronée, d'une autre venant d'un type d'architecture réseau différent ?

Face à cette problématique, nous proposons `RADAR` [1], une nouvelle architecture pour le CS-FL qui garantit la qualité des contributions sans nécessiter d'hypothèse préalable sur l'hétérogénéité des données. Les

‡Co-auteurs principaux ; ils ont contribué de manière équivalente à ce travail.

Les auteurs remercient Hélène Le Bouder pour son encadrement. Ce travail est soutenu par (1) la chaire CyberCNI.fr avec le soutien de l'Union européenne à travers le Fonds européen de développement régional (FEDER) de la Région Bretagne (EU001043), (2) le projet Beyond5G financé par la BPI dans le cadre du plan d'investissement *France Relance* et (3) le Luxembourg National Research Fund (FNR) (référence C23/IS/18088425/COCTEL).

approches existantes dans la littérature reposent généralement sur une source unique de vérité, mais celle-ci est impossible à construire dans des contextes hétérogènes. D'autres cherchent alors à regrouper les participants similaires de sorte à éliminer ceux qui ne le sont pas (*i.e.*, les participants légitimes se ressemblent) ou, au contraire, à écarter les participants similaires considérés comme des attaquants en collusion. Pour s'affranchir de sources de vérité centralisées et d'hypothèses trop fortes sur les attaquants, `RADAR` combine trois briques complémentaires : (*i*) des évaluations croisées émises par les participants sur les contributions de leurs pairs, (*ii*) un algorithme de partitionnement et de regroupement des participants sur la base de leur similarité perçue, et (*iii*) un système de réputation à même de pondérer les contributions avant l'agrégation.

Le présent article propose une synthèse des contributions de `RADAR` :

1. un processus et une architecture pour l'apprentissage fédéré garantissant la qualité du modèle agrégé pour chacun des participants ;
2. une analyse montrant que les métriques d'évaluations de l'apprentissage peuvent être utilisées pour estimer la similarité entre des distributions de données ; et enfin
3. une validation empirique de la combinaison du partitionnement et du système de réputation pour améliorer la qualité des contributions en CS-FL.

2 Architecture

L'architecture de `RADAR` repose sur la complémentarité entre évaluation croisée, partitionnement et système de réputation, permettant ainsi de prendre des décisions de confiance sans recourir à une source de vérité unique. La fig. 1 illustre le processus d'apprentissage et les interactions entre les différentes briques de l'architecture. En particulier, on y observe l'utilisation des évaluations croisées pour le partitionnement, puis l'utilisation des groupes ainsi formés par le système de réputation et l'agrégation.

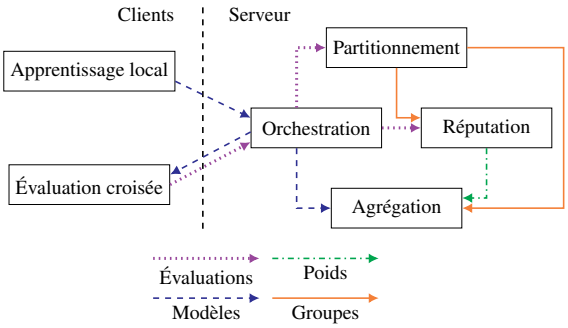

FIGURE 1 : *Panorama de l'architecture.*

2.1 Évaluation croisée des contributions

L'évaluation croisée repose sur le constat que des modèles statistiquement éloignés peuvent être de bonne qualité. Cependant, puisqu'il est impossible de construire une source de vérité unique dans un contexte non indépendant ou identiquement distribué (non-IID), `RADAR` délègue l'évaluation des modèles aux clients. Pour cela, nous ajoutons une étape supplémentaire au processus de FL classique [2], placée après l'entraînement local, lors de laquelle le serveur central sérialise l'ensemble des modèles locaux obtenus et les transmet aux clients. Ceux-ci utilisent un jeu d'évaluation pour mesurer la qualité de chacun des modèles locaux, puis transmettent ces évaluations au serveur. Le surcoût apporté par cette étape est conséquent (*i.e.*, $O(n^2)$ à l'échelle du système et $O(n)$ point de vue des participants), mais nous le considérons acceptable dans le cadre du CS-FL où les participants sont peu nombreux et sans contraintes de ressources. De même, les préoccupations sur la confidentialité des données peuvent être atténuées par l'usage des enclaves sécurisées ou de la confidentialité différentielle. La version complète de `RADAR` discute plus amplement les avantages et inconvénients de cette approche [1].

2.2 Combattre l'hétérogénéité grâce au partitionnement

L'objectif du partitionnement est de regrouper les participants similaires dans des sous-fédérations plus homogènes, lorsque cela est approprié. Contrairement aux approches existantes (*e.g.*, `FoolsGold` [3] ou `FLAME` [4]), `RADAR` compare les résultats des évaluations croisées plutôt que les modèles locaux. Il regroupe ainsi les participants sur leur perception de la qualité des modèles plutôt que sur la distribution de leurs données, introduisant ainsi une forme de subjectivité. Nous utilisons la similarité cosine des évaluations émises envers les autres participants, mettant ainsi l'accent sur les différences de points de vue : une

variation sur un seul modèle peut suffire à séparer deux participants. Le partitionnement des participants est hiérarchique, ce qui évite d'avoir à connaître *a priori* le nombre de sous-fédérations. Ce mode de partitionnement fusionne en effet de manière itérative les groupes les plus proches, jusqu'à atteindre un seuil de distance. `RADAR` utilise un seuil dynamique basé sur la distance moyenne entre les groupes au lancement du processus. Le résultat du partitionnement est utilisé directement par l'agrégation (*c.f.* Figure 1) : un modèle différent est construit pour chaque groupe à partir des modèles locaux de ses participants.

2.3 Contributions de qualité grâce à la réputation

Le système de réputation pondère ensuite les modèles locaux des participants lors de leur agrégation au sein de chaque groupe en utilisant une distribution de Dirichlet multivaluée. Afin d'éviter que des participants en collusion ne manipulent leurs évaluations pour augmenter mutuellement leurs scores ou réduire celui d'autres participants, les évaluations reçues sont pondérées en fonction de leur similarité avec celles des autres membres du groupe avant d'être prises en compte. De plus, l'historique des évaluations passées est pris en compte à l'aide d'une fonction de fraîcheur qui fait décroître de manière exponentielle les scores des rondes précédentes. `RADAR` évite ainsi l'effet d'attaques étalées sur plusieurs rondes tout en permettant à des participants de rectifier leur comportement pour reconstruire leur réputation après avoir été pénalisés. Par construction, notre architecture peut traiter une contribution de mauvaise qualité de deux manières : (*i*) en le plaçant dans un groupe différent, ce qui neutralise son effet sur les autres participants, ou (*ii*) en diminuant son poids dans l'agrégation grâce au système de réputation.

3 Évaluation

`RADAR` a plusieurs objectifs : (*i*) maintenir un haut niveau de performance face à des données hétérogènes, (*ii*) identifier et pénaliser des contributions de faible qualité, et (*iii*) atténuer les attaques d'inversion de labels. En conséquence, l'évaluation repose sur trois catégories de scénarios, `Bénin`, `Unique` et `Collusion`, qui permettent de simuler différents types de comportements *Byzantins* en fonction du nombre de contributions problématiques. Plusieurs paramètres sont testés dans chaque catégorie, comme le type [†] et l'intensité des modifications apportées aux données ou leur évolution dans le temps. L'hétérogénéité est simulée grâce à plusieurs jeux de données de détection d'intrusion sur des traces réseau dont les caractéristiques ont été homogénéisées, formant des cas d'usage différents pour des groupes de participants. La méthodologie est plus amplement détaillée dans le papier original [1] et en annexe de celui-ci. La Table 1 compare `RADAR` avec des solutions de la littérature sur différents scénarios visant à émuler les objectifs visés, en utilisant le taux de réussite de l'attaque (TRA) comme métrique [‡].

3.1 Hétérogénéité

Le scénario `Bénin` présenté en Table 1 permet d'évaluer l'efficacité de notre approche dans un scénario de données hétérogènes exempt de toute attaque. On y observe que `RADAR`, noté `RA` dans la table, présente une excellente exactitude moyenne. Ce résultat sous-tend que les participants similaires sont correctement regroupés ensemble, validé par de plus amples tests [1]. `FoolsGold` [3], noté `FG` dans la table, en revanche, présente de piètres résultats. Cela est dû au fait que l'approche, qui vise à se prémunir d'attaquants en collusion, identifie les participants similaires comme des attaquants et les pénalise.

3.2 Qualité des contributions

Nous évaluons les problématiques de qualité des contributions dans les scénarios marqués `Unique`, où un seul participant présente des données empoisonnées. On constate en Table 1 que le TRA du scénario `Unique` est semblable au `Bénin`, car le poids du participant incriminé a été ajusté par le système de réputation pour compenser la contribution de mauvaise qualité. Les limites structurelles d'approches telles que `FoolsGold` dans les scénarios hétérogènes sont à nouveau mises en avant : l'attaquant seul est considéré comme unique source de vérité, entraînant un TRA supérieur aux approches naïves telles que `FedAvg` [2].

†. Les attaques ciblées concernent des modifications concentrées sur une seule étiquette, là où les attaques non ciblées touchent uniformément les différentes classes.

‡. *I.e.*, taux de faux négatifs sur une étiquette donnée pour les attaques ciblées et 1 − exactitude sinon.

TABLE 1 : *Effet de différentes configurations d'attaques sur les approches de référence.* Le taux de réussite de l'attaque (TRA) est calculé sur la classe affectée pour les attaques ciblées et pour l'ensemble des données dans les attaques non ciblées. `RA` est `RADAR`, `FG` est `FoolsGold`, `FA` est `FedAvg` (sur *tous* les participants), et `FC` est `FedAvg` avec un partitionnement idéal suivant les jeux de données. Le scénario limitant de `RADAR` est marqué ‡.

Scenario	Accuracy moyenne (%)				TRA (%)			
	RA	FG	FA	FC	RA	FG	FA	FC
Ciblé								
Bénin	99.07	55.04	79.49	**99.24**	**0.00**	5.17	5.10	0.09
Unique	99.06	60.51	77.38	**99.22**	**0.00**	93.82	6.73	0.45
Collu. min.	**98.96**	54.64	78.48	98.33	**0.00**	2.97	9.99	53.40
‡ Collu. maj.	**98.28**	85.10	79.40	98.22	73.39	**8.10**	17.65	59.36
Non ciblé								
Bénin	99.07	55.04	79.49	**99.24**	0.09	0.39	33.30	**0.06**
Unique	98.96	49.56	78.38	**99.22**	**0.08**	99.89	54.70	0.12
Collu. min.	**98.98**	49.67	72.47	97.69	0.10	**0.04**	44.53	6.26
Collu. maj.	**98.96**	69.09	81.87	75.66	**0.08**	38.98	59.49	94.36

3.3 Résistance à l'empoisonnement

Des fautes synchonisées sont probablement le résultat d'attaquants en collusion, ce que nous modelisons dans la dernière catégorie de scénarios, `Collusion`. On observe en Table 1 que les attaques non ciblées échouent, *quelle que soit la proportion d'attaquants*. En effet, les perpétrateurs de ces attaques peu discrètes sont isolés par l'algorithme de partitionnement, les rendant inopérants. Dans le cas d'attaques ciblées, attaquants et bénins, trop similaires, sont regroupés et c'est le système de réputation qui arbitre. Par conception, celui-ci favorise la majorité, annulant ainsi l'effet des attaquants minoritaires, mais permettant à ceux majoritaires d'empoisonner le modèle. Dans ce cas spécifique, `FoolsGold` détecte correctement les attaquants en collusion et présente donc les meilleurs résultats des approches testées. L'effet du système de réputation sur les poids des participants du groupe empoisonné est détaillé en Figure 2. On y observe que le système est capable de faire évoluer avec précision le poids des participants qui changent de comportement au cours du temps.

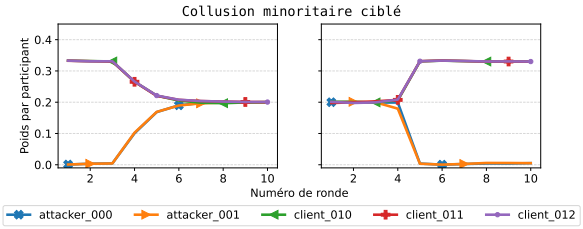

FIGURE 2 : *Poids d'agrégation à chaque ronde pour les participants du groupe empoisonné.* À gauche, les attaquants empoisonnent initialement leurs données avant de devenir `Bénin` à la ronde 3 : ils sont progressivement pardonnés. À droite, les attaquants sont initialement `Bénin` puis empoisonnent progressivement leurs données : leur poids se dégrade rapidement.

4 Conclusion et discussion

La version originale distingue les effets du partitionnement et ceux de la réputation, et teste davantage de scénarios en faisant varier le taux de données empoisonnées [1]. Ces expérimentations permettent de conclure que `RADAR` produit d'excellents résultats dans une majorité de scénarios. Bien que les expérimentations aient été réalisées sur des données de détection d'intrusion, `RADAR` peut être appliqué à d'autres cas d'usages pourvu que (*i*) ces cas d'usages utilisent des modèles paramétriques compatibles avec le FL, et (*ii*) les participants disposent de données locales leur permettant d'évaluer les modèles des autres participants. Réduire le coût de l'évaluation croisée reste un défi majeur pour l'adoption de `RADAR` dans des systèmes à grande échelle, où le nombre de participants est élevé et où les ressources sont limitées. Des mécanismes d'agrégation décentralisée, tels que les algorithmes par commérage, pourraient également être envisagés afin de s'affranchir du serveur central.

Références

[1] L. LAVAUR, P.-M. LECHEVALIER, Y. BUSNEL, R. LUDINARD, M.-O. PAHL et G. TEXIER, RADAR: Model Quality Assessment for Reputation-aware Collaborative Federated Learning. In *2024 43rd International Symposium on Reliable Distributed Systems (SRDS)*, 2024, p. 222-234.

[2] B. MCMAHAN, E. MOORE, D. RAMAGE, S. HAMPSON et B. A. y. ARCAS, Communication-efficient learning of deep networks from decentralized data. In *20th international conference on artificial intelligence and statistics*, 2017.

[3] C. FUNG, C. J. M. YOON et I. BESCHASTNIKH, The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses*, sér. RAID, 2020.

[4] T. D. NGUYEN, P. RIEGER, H. CHEN et al., FLAME: Taming Backdoors in Federated Learning. In *31st USENIX Security Symposium (USENIX Security 22)*, 2022.

[5] S. I. POPOOLA, G. GUI, B. ADEBISI, M. HAMMOUDEH et H. GACANIN, Federated Deep Learning for Collaborative Intrusion Detection in Heterogeneous Networks. In *IEEE 94th Vehicular Technology Conference*, sér. VTC2021-Fall, 2021.

A Disponibilité et reproductibilité

Bien que les informations nécessaires à la reproductibilité de `RADAR` soient déjà disponibles dans le papier original [1] et sur son dépôt GitHub [§], nous détaillons ci-après les différents éléments à prendre en compte pour reproduire au mieux et éventuellement étendre les résultats présentés.

Algorithme et jeux de données L'évaluation présentée repose sur uniquement sur des algorithmes issus de la littérature. Ainsi, le modèle utilisé est un perceptron à deux couches cachées, tel qu'implémenté par POPOOLA et al. [5]. Les jeux de données utilisés sont une collection de jeux de données publics (CSE-CIC-IDS2018, Bot-IoT, ToN_IoT et UNSW-NB15), tous constitués de flux réseaux pour la détection d'intrusions. Réalisées par l'Université de Queensland (Australie), ces versions possèdent des caractéristiques homogènes, permettant leur usage dans des contextes fédérés. Ils sont disponibles en téléchargement sur leur site internet [¶]. Les paramètres de l'algorithme de partitionnement et de réputation sont détaillés dans le papier original [1].

Gestion des dépendances Afin de garantir au mieux l'environnement système et logiciel de l'expérimentation, `RADAR` est implémenté en Python avec Poetry [‖] pour la gestion des dépendances. Poetry est un solveur de dépendances et un gestionnaire de paquet qui maintient son état à l'aide d'un fichier de verrouillage (*lock file*), permettant à un utilisateur ultérieur de télécharger les mêmes versions des logiciels, à l'octet prêt. Construit autour de principes similaires, l'environnement système est géré par Nix [**], garantissant que les dépendances en dehors de l'environnement Python sont, elles aussi, verrouillées.

Implémentation logicielle `RADAR` est implémenté à l'aide de Flower [††] dans sa version 1.4.0. Flower est une suite d'outils pour orchestrer ou simuler des apprentissages fédérés. Notez que `RADAR` vient avec une version alternative de certaines fonctionnalités de Flower pour implémenter son architecture. Il est donc nécessaire d'utiliser aussi les modules Python de substitution fournis dans le dépôt GitHub pour reproduire le déroulement opérationnel de `RADAR`.

§. `https://github.com/leolavaur/radar-srds-2024`
¶. `https://staff.itee.uq.edu.au/marius/NIDS_datasets/`
‖. `https://python-poetry.org/`
**. `https://nixos.org`
††. `https://flower.ai/`