Apprentissage Fédéré pour la détection des intrusions

par

Mohamed Ali AYED

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE LOGICIEL
M. Sc. A.

MONTRÉAL, LE 5 FÉVRIER 2022

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Chamseddine Talhi, directeur de mémoire
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Ali Ouni, président du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Kaiwen Zhang, examinateur externe
Département de génie logiciel et des TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE "DATE DE SOUTENANCE"

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

À mes bien-aimés parents,

Tous les mots du monde ne sauraient exprimer l'immense amour que je vous porte, ni la profonde gratitude que je vous témoigne pour tous les efforts et les sacrifices que vous n'avez jamais cessé de consentir pour mon instruction et mon bien-être, vous représentez les personnes les plus précieuses à mes yeux.

J'espère avoir répondu aux espoirs que vous avez fondés en moi.

Je vous rends hommage par ce modeste travail en guise de ma reconnaissance éternelle et de mon infini amour.

Vous résumez si bien le mot parent qu'il serait superflu d'y ajouter quelque chose.

Je tiens aussi à remercier mon directeur de recherche Chamseddine Talhi, pour m'avoir offert l'inestimable opportunité de mener mes recherches en m'accueillant sous ses ailes. Pour sa patience, pour son assistance ininterrompue et ses conseils judicieux tout au long de ce projet qui ont rendu les deux dernières années si fructueuses et éclairantes.

J'exprime ma gratitude à Mitacs pour le financement de ce travail.
Je remercie Ericsson pour leurs soutien durant ma période de maîtrise.
Je remercie sincèrement les membres du jury d'avoir pris le temps de revoir mon mémoire et pour leur précieux retour d'information.

Je tiens également à remercier l'ETS pour son souci de procurer à ses étudiants de telles opportunités les encourageant à se confronter de manière progressive à la vie professionnelle.

Apprentissage Fédéré pour la détection des intrusions

Mohamed Ali AYED

RÉSUMÉ

Avec la croissance rapide des technologies de l'information, en particulier des formes de communication et des médias sociaux, un nombre toujours croissant d'inventions sont interconnectées à davantage d'appareils, ce qui peut conduire à la divulgation d'informations précieuses. Par conséquent, toute personne utilisant un appareil associé à Internet est vulnérable face aux dangers que représentent les prédateurs en ligne. Les problèmes de cybersécurité constituent un combat quotidien face à l'apparition d'attaques toujours plus agressives et brutales. De nombreux appareils ont du mal à programmer les mises à jour automatiques; ils manquent de réactivité comme le recours aux systèmes de détection d'intrusion (IDS) pour déceler toute vulnérabilité. Par conséquent, une nouvelle approche de détection d'intrusion s'impose comme un réel besoin en matière de cybersécurité permettant également d'appréhender les attaques cybernétiques sophistiquées les moins prédictibles.

Mon mémoire s'intéresse à ce domaine. En effet, il porte sur une forte implication des techniques modernes de l'intelligence artificielle pour la détection d'intrusions et plus précisément la classification des attaques réseau. Ainsi, ce rapport fournit des réalisations dont le but est de procurer de nouvelles alternatives en matière de détection d'attaques grâce à l'apprentissage fédéré qui va permettre de préserver les informations sensibles des utilisateurs. Les objectifs de ce mémoire sont de rechercher, collecter des jeux de données, préparer des données, modéliser et évaluer des modèles fédérés pour la détection d'intrusions réseau.

Nous avons réussi à implémenter l'apprentissage fédéré avec deux types de classifications. Une classification binaire utilisant un modèle réseau neuronal convolutif fédéré sur le jeu de données IDS2017 et deux classifications multi-classe avec un perceptron multicouche fédéré sur le jeu de données IDS2018. Pour ces deux types de classifications, quatre stratégies du choix du client ont été utilisé. Tous les clients sont sélectionnés, les clients sont choisis au hasard, sélection en fonction des types d'attaques ou en fonction des systèmes d'exploitation pendant un tour.

les résultats obtenus pour la classification binaire avec les quatre stratégies ont bien fonctionné avec une précision allant de 87% jusqu'à 93% pour la stratégie où tous les clients sont sélectionnés et en foncion des types d'attaques . Pour la classification multi-classe le meilleur résultat obtenu est avec la sélection de tous les clients avec une précision de 90%.
Cela montre même dans un environnement réel où des machines qui tombent en panne ou des coupures de courant peuvent survenir, l'apprentissage fédéré reste efficace et robuste dans ces situations.

Mots-clés: apprentissage fédéré, IDS

Federated learning for intrusion detection

Mohamed Ali AYED

ABSTRACT

With the rapid growth of information technology, especially the expansion of communication and social media, an ever-increasing number of devices are interconnected, this can lead to the disclosure of sensitive and private information. Therefore, anyone who uses a device associated with the internet is helpless in the face of the dangers presented by online hackers. Cybersecurity is becoming a daily struggle with the emergence of new attacks. Many devices find it difficult to schedule automatic updates ; they lack responsiveness measures like intrusion detection systems (IDS) to detect vulnerability Therefore, there is a need for a new intrusion detection approach that will provide security and can also prevent new intrusion attacks.

My project is addressing this problem. Indeed, it relates to the strong implication in the use of modern techniques of artificial intelligence for the intrusion detection the classification of the network attacks. Thus, this report explains the achievements of a new alternative method to the detection of attacks through federated learning. The objectives to be achieved during this stage are research, dataset collection, data preparation, modeling and evaluation of federated models for network intrusion detection.

We successfully implemented federated learning with two types of classifications. A binary classification using a federated convolutional neural network model on the IDS2017 dataset and two multi-class classifications with a federated multilayer perceptron on the IDS2018 dataset. For these two types of classifications, four strategies of the client's choice were used. All clients are selected, clients are randomly selected, selection based on attack types or based on operating systems during a round.

the results obtained for the binary classification with the four strategies worked well with an accuracy ranging from 87% up to 93% for the strategy where all customers are selected and depending on the types of attacks. For the multi-class classification the best result obtained is with the selection of all clients with an accuracy of 90%.
This shows even in a real-world environment where machines crashing or power outages can occur, Federated Learning remains effective and robust in these situations.

Keywords: federated learning, IDS

TABLE DES MATIÈRES

Page

INTRODUCTION 1

CHAPITRE 1 NOTIONS DE BASE ET TRAVAUX CONNEXES 5
1.1 Introduction 5
1.2 Notions de base 5
1.2.1 Critères de détection d'intrusions réseau 5
1.2.1.1 Systèmes de détection d'intrusion par signatures 5
1.2.1.2 Systèmes de détection d'intrusion par anomalies 6
1.2.2 Attaques sur les réseaux informatiques 7
1.2.2.1 Attaques par déni de service 7
1.2.2.2 Attaques par force brute 8
1.2.2.3 Attaques Heartbleed 8
1.2.3 Apprentissage automatique 8
1.2.3.1 Types d'apprentissage 8
1.2.3.2 Le jeu de données 10
1.2.3.3 Évaluation du modèle 10
1.2.4 Réseau de neurones 11
1.2.4.1 Un neurone 11
1.2.4.2 Les couches 12
1.2.5 Apprentissage fédéré 13
1.2.5.1 Sélection des clients 16
1.2.5.2 Hyperparamètre de l'apprentissage fédéré 16
1.3 Travaux connexes 17
1.3.1 Obtention des données 17
1.3.2 Détection d'intrusion avec l'apprentissage automatique 19
1.3.3 Détection d'intrusion avec l'apprentissage fédéré 20
1.3.4 Lacune des travaux existants 22
1.4 Conclusion 23

CHAPITRE 2 SOLUTION PROPOSÉE 25
2.1 Introduction 25
2.2 L'approche de l'apprentissage fédéré 25
2.3 Création d'un ensemble de données pour la détection d'intrusion réseau basée sur l'apprentissage fédéré 26
2.4 Ensemble de données : IDS 2017 27
2.5 Ensemble de données : IDS 2018 32
2.6 Conclusion 37

CHAPITRE 3 EXPÉRIMENTATION ET ÉVALUATION 39
3.1 Introduction 39

3.2 Expérimentation 39
3.2.1 Le fractionnement du jeu de données 39
3.2.2 Métrique : la perte 40
3.2.3 Métrique : la précision 40
3.2.4 Métrique : sensibilité et spécificité 41
3.3 Expérimentation : IDS 2017 42
3.3.1 Tous les clients sont sélectionnés 42
3.3.2 Clients aléatoires sélectionnés 43
3.3.3 Sélection des clients en fonction des attaques 44
3.3.4 Sélection du client en fonction du système d'exploitation 46
3.3.5 Comparaison entre les quatre approches 48
3.4 Expérimentation : IDS 2018 49
3.4.1 Tous les clients sont sélectionnés 49
3.4.2 Clients aléatoires sélectionnés 51
3.4.3 Sélection des clients en fonction des attaques 52
3.4.4 Sélection en fonction du système d'exploitation 54
3.5 Conclusion 55

CONCLUSION ET RECOMMANDATIONS 57

ANNEXE I IMPLÉMENTATION 59

ANNEXE II FEDERATED LEARNING FOR ANOMALY-BASED INTRUSION DETECTION 61

BIBLIOGRAPHIE 69

LISTE DES TABLEAUX

Page

Tableau 2.1 Étiquette des données par jour 29

Tableau 2.2 Dataset IDS2017 - nombre de paquets par attaque 30

Tableau 2.3 Distribution des instances de l'IDS2017 sur les Clients FL 31

Tableau 2.4 Caractéristiques de la machine 34

Tableau 2.5 Distribution des instances de l'IDS2018 sur les Clients FL 36

LISTE DES FIGURES

Page

Figure 1.1 Attaque DDoS....7

Figure 1.2 Apprentissage automatique, tirée du conseil de l Europe (2020)....9

Figure 1.3 Jeu de données, tirée de Azencott (2020) 10

Figure 1.4 Évaluation du modèle, tirée de Maurice (2018)....11

Figure 1.5 Un neurone, tirée de Kang (2017)) 12

Figure 1.6 Un réseau de neurones, tirée de Raycad (2017) 13

Figure 1.7 Apprentissage fédéré, tirée de Zhu (2020)....14

Figure 1.8 Choix du modèle, tirée de Subramaniam (2019a)) 14

Figure 1.9 Diffusion du modèle, tirée de Subramaniam (2019a) 15

Figure 1.10 Entraînement du modèle par les périphériques, tirée de Subramaniam (2019b) 15

Figure 1.11 Agrégation des gradients, tirée de Subramaniam (2019c) 16

Figure 1.12 Architecture réseau du CICIDS2017, tirée de Sharafaldin, Habibi Lashkari & Ghorbani (2018)....18

Figure 2.1 Architecture de l'apprentissage fédéré....26

Figure 2.2 Réseau IDS 2017, tirée de Sharafaldin *et al.* (2018)....28

Figure 2.3 Réseau IDS 2018, tirée de of new brunswick (2018)....33

Figure 3.1 Répartition du jeu de données....39

Figure 3.2 Sensibilité et spécificité, tirée de Datamok (2021) 41

Figure 3.3 Visualisation de la perte de l'apprentissage sur tous les clients....43

Figure 3.4 Visualisation de la précision de l'apprentissage sur tous les clients 43

Figure 3.5 Visualisation de la perte de l'apprentissage sur des clients aléatoires 44

Figure 3.6 Visualisation de la précision de l'apprentissage sur des clients aléatoires 44

Figure 3.7 Visualisation de la perte de l'apprentissage sur les clients en fonction des attaques 45

Figure 3.8 Visualisation de la précision de l'apprentissage sur les clients en fonction des attaques 46

Figure 3.9 Visualisation de la perte de l'apprentissage sur les clients en fonction des systèmes d'exploitation 47

Figure 3.10 Visualisation de la précision de l'apprentissage sur les clients en fonction des systèmes d'exploitation 47

Figure 3.11 Comparaison entre le taux de précision et la perte 48

Figure 3.12 Décentralisé vs centralisé 49

Figure 3.13 Visualisation de la perte de l'apprentissage sur tous les clients IDS2018 50

Figure 3.14 Visualisation de la précision de l'apprentissage sur tous les clients IDS2018 50

Figure 3.15 Visualisation de la perte sur les clients aléatoires IDS2018 51

Figure 3.16 Visualisation de la précision de l'apprentissage sur les clients aléatoires IDS2018 52

Figure 3.17 Visualisation de la perte sur les clients en fonction des attaques IDS2018 53

Figure 3.18 Visualisation de la précision de l'apprentissage sur les clients en fonction des attaques IDS2018 53

Figure 3.19 Visualisation de la perte sur les clients en fonction des systèmes d'exploitation IDS2018 54

Figure 3.20 Visualisation de la précision de l'apprentissage sur les clients en fonction des systèmes d'exploitation IDS2018 55

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ETS	École de Technologie Supérieure
ML	Apprentissage automatique (Machine learning)
FL	Apprentissage fédéré (Federated learning)
IDS	Systèmes de détection d'intrusion(Intrusion Detection Systems)
CNN	Réseau neuronal convolutif (Convolutional neural network)
MLP	Perceptron multicouche
ANN	Réseau neuronal artificiel (Artificial Neural Network)
DIADEM	Diagnostic des modèles de détection (DIAgnosis of DEtection Models)

INTRODUCTION

De nos jours, des milliers d'internautes sont connectés et s'échangent des messages pouvant être la cible d'attaques. La sécurité du réseau entre en jeu pour éviter que le réseau ne soit endommagé lors de différents assauts comme des attaques par force brute, DDos. Pour pallier ces vulnérabilités, les systèmes de détection d'intrusion sont considérés comme une solution efficace permettant la détection des intrusions en analysant le trafic réseau. Avec la découverte de nouvelles attaques et la complexité de les repérer, les méthodes classiques de détection des attaques sur les réseaux deviennent obsolètes. Les entreprises ont été contraintes de faire appel à des technologies modernes aux résultats encourageants pour une meilleure efficacité en matière de suivi et de prévention des attaques réseau. C'est dans ce contexte que des systèmes utilisant des méthodes d'apprentissage automatique ont vu le jour afin de pallier ce problème.

Cependant, ces systèmes de détection d'intrusions présentent un inconvénient. Afin de détecter les intrusions, ces derniers nécessitent des jeux de données contenant des informations sensibles d'utilisateurs, ce qui peut porter atteinte à leurs vies privées. Les questions de recherche spécifiques de cette recherche sont les suivantes :

- Comment détecter les intrusions de cybersécurité et les classer dans une catégorie appropriée avec une précision supérieure ou égale à l'apprentissage centralisé ?
- Comment entraîner un système de détection d'intrusion en évitant de collecter des informations sensibles sur d'autres machines ?

Ce travail fait face à des défis qui sont traités à partir de perspectives multiples : Préparation des données, prédiction des anomalies avec une précision comparable à celle de l'approche centralisée tout en respectant la vie privée des utilisateurs.

Dans cette recherche, nous ambitionnons de présenter une nouvelle méthode de détection et de classification des intrusions réseau. Les attaques réseau peuvent induire des dommages

notables sur divers systèmes. L'utilisation de méthodes formelles et de méthodes d'apprentissage en profondeur nous aidera à identifier les intrusions en temps réel afin de les contrer. Cela permettra de reconnaître les intrusions avant qu'elles ne causent des dommages considérables. De plus, nous assurerons la classification des intrusions pour une identification et reconnaissance des catégories et types d'intrusions concomitantes. Cela permettra de déterminer la stratégie appropriée pour contrer l'intrusion.

Nous présenterons une méthode de détection et de classification des intrusions réseau utilisant différents modèles avec deux jeux de données, dans le but d'atteindre une précision supérieure ou égale à celle de l'apprentissage centralisé, tout en respectant la vie privée des utilisateurs.

Nous nous sommes fixés les objectifs spécifiques suivants :

- Préparer un ensemble de données adapté pour un apprentissage fédéré.
- Entraîner un classifieur binaire utilisant un modèle réseau neuronal convolutif CNN fédéré sur le jeu de données IDS2017.
- Entraîner un classifieur multiclasse pouvant prédire le type d'attaque avec un modèle perceptron multicouche MLP fédéré sur le jeu de données IDS2018.
- Faire la comparaison des résultats obtenus avec l'apprentissage centralisé.

Une revue de la littérature sur les processus utilisés pour les systèmes de détection d'intrusion par apprentissage automatique s'avère un bon départ pour atteindre ces objectifs. Grâce à ces informations, un système de détection profitable aux entreprises a pu être produit afin d'assurer une protection contre les attaques malveillantes en recourant à une approche distribuée. Ce mémoire, comporte une phase de préparation des jeux de données, des stratégies de sélection de clients en fonction du type d'attaque, du système d'exploitation et cela de manière aléatoire afin de simuler un comportement plus réel. Il a également été fait recours à la prédiction et a l'identification des anomalies grâce à l'apprentissage fédéré.

Ce mémoire apportera ainsi une contribution aux travaux de recherches scientifiques en cybersécurité afin de construire des modèles fédérés sans la nécessité de collecter des informations personnelles propres aux utilisateurs.

Pour bien mener notre travail, nous avons divisé ce mémoire en trois chapitres. Le premier chapitre présentera les notions de base où deux méthodes de classification d'intrusion, par signature et par anomalie, seront exposées. Par la suite, nous définirons quelques types d'attaques parmi les plus connus et les plus couramment utilisés dans les jeux de données sélectionnés. Afin de choisir un modèle d'apprentissage adéquat, nous devons parfaitement discerner les catégories d'apprentissage, la division des jeux de données et l'évaluation de ces derniers. Enfin, nous clôturerons le chapitre par une définition des notions de base de l'apprentissage fédéré. Il comprendra aussi les travaux connexes à partir desquels nous nous sommes inspirés. Dans cette partie, les articles scientifiques traitant des problématiques similaires à la nôtre seront discutés et nous commenterons l'article abordant le jeu de données qui a été sélectionné ainsi que les résultats qu'il a atteints avec l'apprentissage centralisé. Par la suite, nous mentionnerons des systèmes de détection d'intrusion tout en démontrant leurs incapacités à détecter les nouvelles attaques.

Le second chapitre se focalisera sur les contributions apportées. Nous discuterons de l'approche adoptée par l'apprentissage fédéré ainsi que son fonctionnement. Par la suite, nous expliquerons la démarche à suivre pour la création d'un jeu de données approprié à l'apprentissage fédéré et nous nous focaliserons sur le prétraitement des données IDS2017 et IDS2018.

Le dernier chapitre sera consacré aux métriques utilisées pour l'évaluation. Nous procéderons, dans cette partie, à l'analyse et l'interprétation des résultats obtenus lors des expérimentations incluant les deux jeux de données ainsi que les stratégies utilisées pour la sélection des clients.

Enfin, nous conclurons notre travail par une récapitulation des différentes parties évoquant nos contributions. Une proposition pour les travaux futurs sera présentée et discutée dans cette conclusion.

CHAPITRE 1

NOTIONS DE BASE ET TRAVAUX CONNEXES

1.1 Introduction

Durant ce chapitre, nous allons tout d'abord introduire les notions de bases des attaques réseau ainsi que de l'apprentissage machine. Par la suite, nous expliquerons davantage l'apprentissage fédéré. Enfin Nous discuterons les travaux connexes.

1.2 Notions de base

Nous nous sommes concentrés sur les termes : attaques sur les réseaux informatiques, apprentissage automatique, réseau de neurones et apprentissage fédéré.

1.2.1 Critères de détection d'intrusions réseau

Un système de détection d'intrusion (ou IDS : Intrusion Detection System) est un mécanisme destiné à détecter une activité anormale fournissant ainsi la connaissance des tentatives réussies telles que les intrusions bloquées.

Il existe deux grandes classes d'IDS, les plus connues étant les détections de signatures et les détections d'anomalies souvent associées à l'apprentissage automatique.

Certains IDS ont la capacité de répondre aux menaces qu'ils ont détectées. Ces IDS réactifs sont des systèmes de prévention d'intrusions.

1.2.1.1 Systèmes de détection d'intrusion par signatures

Les systèmes de détection d'intrusion basés sur les signatures (SIDS) utilisent des bibliothèques appelées signatures. Lors de l'analyse des flux réseau, le système analyse chaque transmission et transmet un message lorsqu'un comportement anormal se produit. Cependant, cette méthode

n'est valable que si les différentes attaques sont parfaitement identifiées et nécessite une base de signatures régulièrement mise à jour. Ce modèle de détection s'avère inefficace et aucune alerte n'est générée dans le cas d'attaques inconnues de la base.

Il existe plusieurs implémentations permettant la reconnaissance de signature, notamment :

- Les règles de filtrage sont représentés sous forme d'arbres de décision, chaque feuille de l'arbre correspondant à une règle.
- Les systèmes de transition d'état décrivent les intrusions comme un scénario étant lui-même représenté comme une série de transitions qui caractérisent l'évolution des états du système.

1.2.1.2 Systèmes de détection d'intrusion par anomalies

Les systèmes basés sur les anomalies, à l'opposé du SIDS, ne reposent pas sur des bibliothèques, mais sont chargés de détecter automatiquement les comportements anormaux à travers l'apprentissage. Ces derniers s'appuient sur deux phases :

- la première étant la phase d'apprentissage, durant laquelle le système étudie le comportement des flux normaux du réseau.
- la seconde étant la phase de détection au cours de laquelle le système analyse le trafic et tente d'identifier les événements anormaux sur la base de cette connaissance.

Cette méthode de détection se fonde sur de nombreuses techniques d'apprentissage supervisé, comme les réseaux de neurones artificiels ou les machines à vecteurs de support.

En 2019, la détection d'attaques anormales par l'intermédiaire des techniques d'apprentissage supervisé a été reconnue par la communauté comme très efficace. En effet, selon la méthode d'apprentissage mise en œuvre, la précision des résultats peut rapidement atteindre plus de 90% d'exactitude. Wang, Zhu, Tian, Xin, Niu & Yang (2011) ; Wang, Wong & Miner (2004) ; Zhang, Xu & Gong (2015).

1.2.2 Attaques sur les réseaux informatiques

"Tout ordinateur connecté à un réseau informatique est potentiellement vulnérable à une attaque. Cette dernière représente l'exploitation d'une faille dans un système informatique pour des raisons inconnues.

Sur Internet, des attaques ont lieu en permanence. Ces attaques sont, pour la plupart, lancées automatiquement à partir de machines infectées (par des virus, chevaux de Troie, vers).

Afin de contrer ces assauts, il est indispensable d'identifier les principaux types d'attaques afin de mettre en œuvre des dispositions préventives." Pillou (2018)

1.2.2.1 Attaques par déni de service

L'attaque par déni de service, également appelé Dos, est une attaque qui a pour but de rendre un service indisponible en l'inondant de transmission réseau. Cependant, cette attaque ne procure aucun avantage à l'attaquant par rapport à d'autres types d'attaques qui permettent d'obtenir ou de faciliter un accès. Si ce type d'attaque provient de plusieurs sources, il est alors appelé déni de service distribué (DDos). On peut observer le fonctionnement de cette attaque dans la figure 1.1.

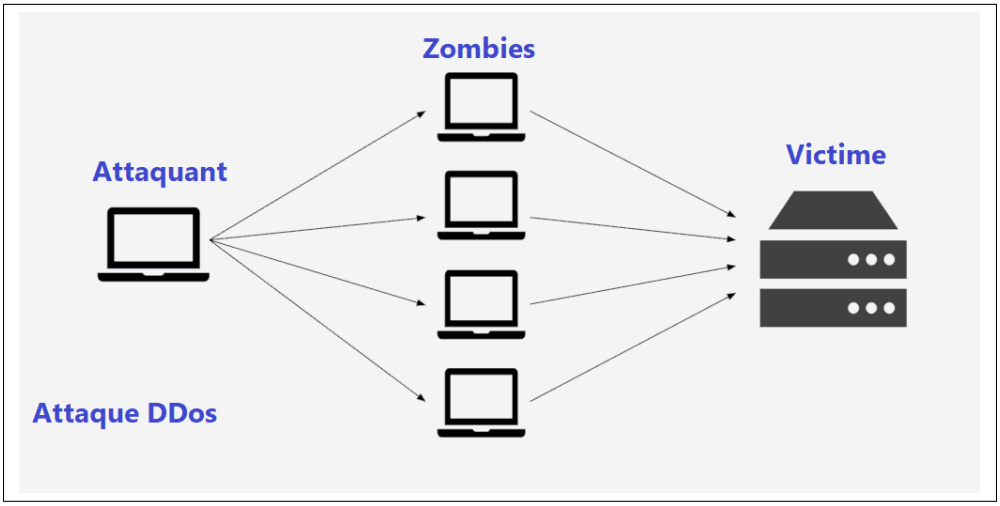

Figure 1.1 Attaque DDoS

1.2.2.2 Attaques par force brute

Cette méthode est la plus ancienne et la plus simple à accomplir. Elle permet de craquer un nom d'utilisateur, une clé de chiffrement, un mot de passe ou même une page Web cachée à travers plusieurs processus d'essai et d'erreur. Cette méthode demeure toujours efficace de nos jours, mais le temps nécessaire pour réussir une attaque dépend de la complexité et de la longueur du contenu à obtenir. Cela peut nécessiter plusieurs secondes à plusieurs années.

1.2.2.3 Attaques Heartbleed

La faille HeartBleed fut introduite par Robin Seggelmann, un étudiant à l'université de Duisburg-Essen, qui a conçu l'extension HeartBeat pour OpenSSL. Cette méthode se base sur le fait que le serveur ne vérifie pas que l'entier envoyé correspond concrètement à la taille du message. De ce fait, l'attaquant pourra introduire un entier supérieur à la taille du message, de telle sorte que le serveur remplit la différence du contenu de sa mémoire qui peut contenir des mots de passe ou encore des clés privées de certification.

1.2.3 Apprentissage automatique

L'apprentissage automatique est un champ d'études de l'intelligence artificielle qui se fonde sur des approches statistiques pour donner aux ordinateurs la capacité d'apprendre à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des problèmes sans être explicitement programmés à le faire.

1.2.3.1 Types d'apprentissage

Ces méthodes sont habituellement classées en trois catégories :

- **L'apprentissage supervisé** est une tâche consistant à apprendre une fonction de prédiction à partir d'exemples annotés. Nous distinguons les problèmes de régression des problèmes de classement. Ainsi, nous considérons que les problèmes de prédiction d'une variable

quantitative sont des problèmes de régression, tandis que les problèmes de prédiction d'une variable qualitative sont des problèmes de classification.

- **L'apprentissage non supervisé** : Il s'agit pour un modèle de trouver des caractéristiques à partir de données non étiquetées. Puisque ces dernières ne sont pas étiquetées, il n'est pas possible d'affecter au résultat de l'algorithme utilisé, un score d'adéquation.
- **L'apprentissage par renforcement** consiste à apprendre les actions à entreprendre à partir d'interactions avec l'environnement en vue d'une optimisation permettant l'atteinte d'une récompense.

Nous présentons dans la figure 1.2 les différents types d'apprentissages :

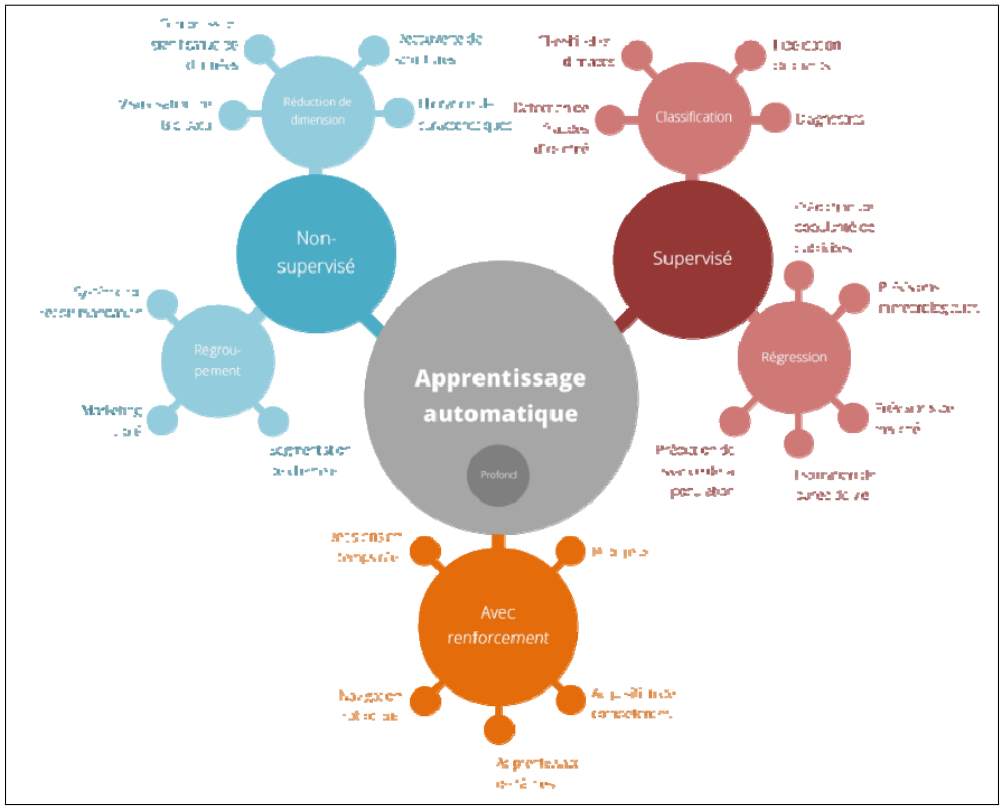

Figure 1.2 Apprentissage automatique, tirée du conseil de l Europe (2020)

1.2.3.2 Le jeu de données

Dans cette section, nous examinerons comment les ensembles de données de l'apprentissage du test et de la validation sont définis et comment ils diffèrent selon les principaux textes et références d'apprentissage supervisé(Figure 1.3).

- Le jeu de données d'apprentissage : l'échantillon de données utilisé pour l'apprentissage du modèle.
- Le jeu de données de la validation : l'échantillon de données utilisé afin d'évaluer le modèle sur les données de l'entraînement, tout en ajustant les hyperparamètres de modèle.
- Le jeu de données du test : l'échantillon de données utilisé pour fournir une évaluation du modèle sur le reste des données qui sont inconnues.

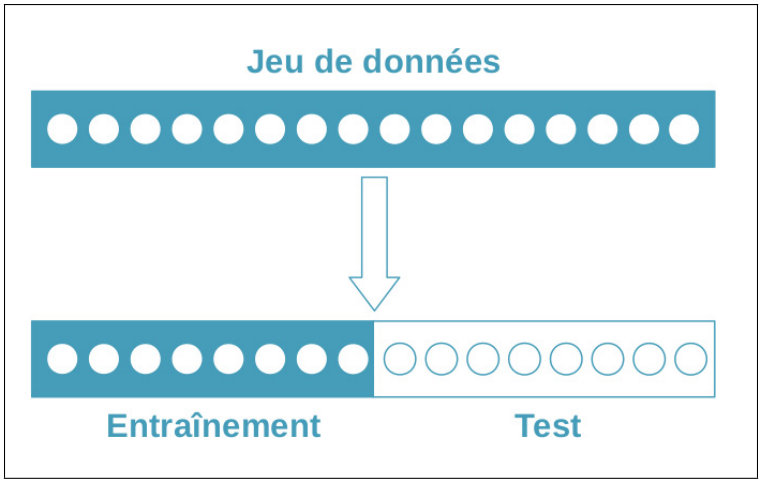

Figure 1.3 Jeu de données, tirée de Azencott (2020)

1.2.3.3 Évaluation du modèle

Durant l'évaluation de l'apprentissage supervisé, nous pouvons retrouver les trois types de modèles suivants(Figure 1.4) :

- **Le sur-apprentissage** (Overfitting) désigne le fait que le modèle prédictif produit par l'algorithme de l'apprentissage automatique s'adapte bien au Training Set.

- **Le sous-apprentissage** (Underfitting)correspond au modèle prédictif qui s'adapte mal et aboutit à de mauvais résultats lors du Training Set.
- **Le juste milieu** (fitting) est un modèle qui ne souffre ni de sur-apprentissage ni de sous-apprentissage. En d'autres termes, il ne souffre ni d'un grand biais ni d'une grande variance.

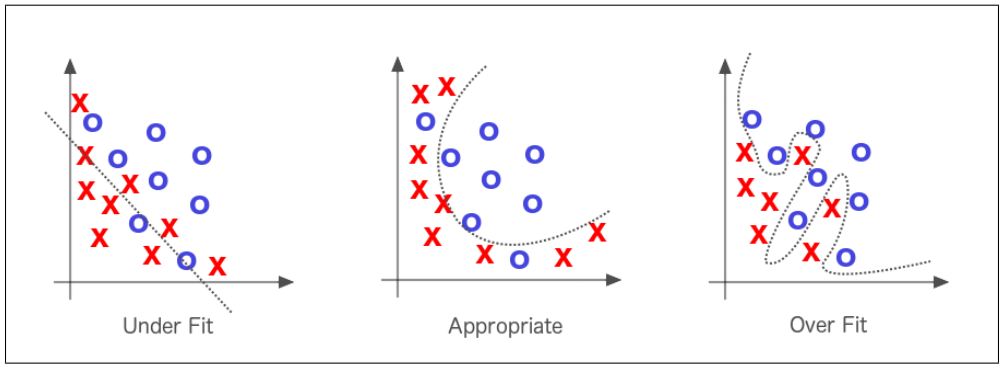

Figure 1.4 Évaluation du modèle, tirée de Maurice (2018)

1.2.4 Réseau de neurones

Dans cette section, nous évoquons deux concepts clés des réseaux de neurones : (1) le neurone qui est l'unité de base du calcul dans un réseau de neurones souvent appelé nœud ou unité et (2) les couches qui structurent un réseau de neurones.

1.2.4.1 Un neurone

Un neurone reçoit des entrées d'autres nœuds ou d'une source externe et calcule une sortie. Chaque entrée a un poids associé w appelé weights, qui est attribué en fonction de son importance relative par rapport aux autres entrées. Le nœud applique une fonction f , pour fonction d'activation, à la somme pondérée de ses entrées. (voir figure 1.5)

Les entrées (input) du neurone sont X_1 et X_2 qui ont les poids (weights) w_1 et w_2. De plus, il existe une autre entrée avec un poids b (biais) qui lui est associée. La fonction principale du Bias est de fournir à chaque nœud une valeur constante pouvant être entraînée et permet de décaler la fonction d'activation à gauche ou à droite.

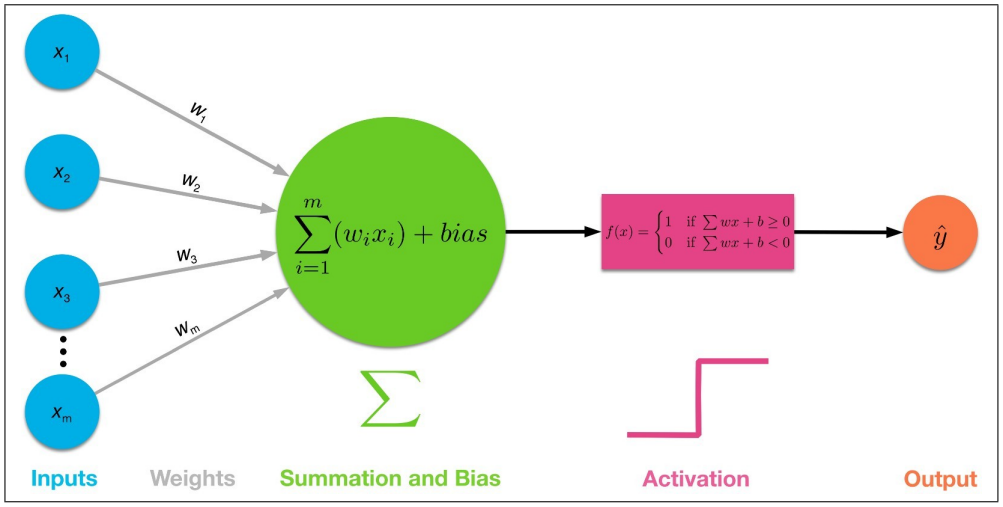

Figure 1.5 Un neurone, tirée de Kang (2017))

La sortie Y du neurone est calculée comme le montre la figure 1.4. La fonction f est non linéaire et appelée fonction d'activation. La fonction d'activation a pour but d'introduire la non-linéarité dans la sortie d'un neurone. Parmi les fonctions d'activation, nous citons sigmoid, tanh et Relu qui est utilisé par notre architecture.

Relu signifie Unité Linéaire Rectifiée qui prend une entrée de valeur réelle avec une valeur minimale 0 (remplace les valeurs négatives par zéro) : $f(x) = max(0, x)$.

1.2.4.2 Les couches

Un réseau de neurones est un ensemble de neurones formels inter-connectés formé de trois couches : couche d'entrée (input layer), couche cachée (hidden layer) et couche de sortie (output layer). Chaque couche est composée d'un ou plusieurs nœuds, représentés dans ce diagramme par les petits cercles. Les arcs entre les nœuds indiquent le flux d'information d'un nœud à l'autre. Dans ce type particulier de réseau de neurones, les informations ne circulent que de l'entrée à la sortie (c'est-à-dire de la gauche vers la droite) (voir figure 1.6). D'autres types de réseaux de neurones ont des connexions plus complexes, telles que des chemins de retour (backpropagation).

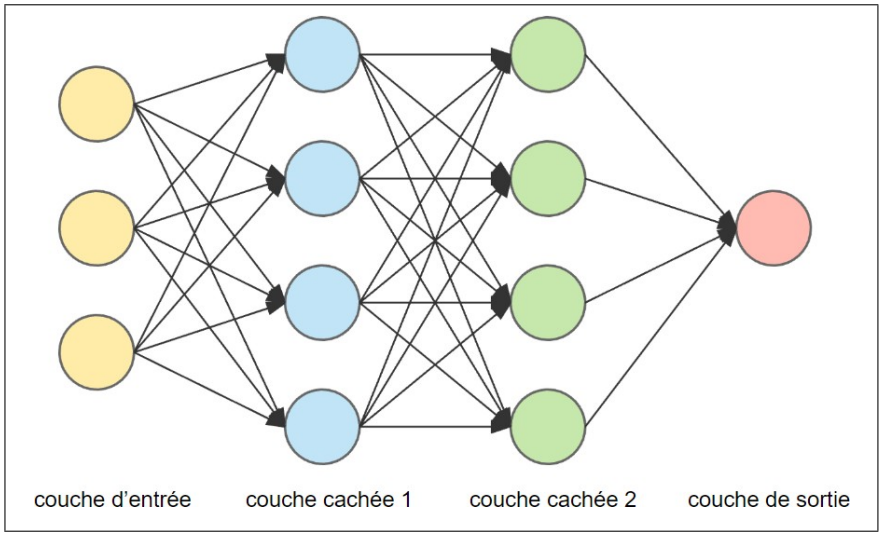

Figure 1.6 Un réseau de neurones, tirée de Raycad (2017)

1.2.5 Apprentissage fédéré

Les approches d'apprentissage automatique standard nécessitent de centraliser les données d'entraînement sur une machine ou dans un serveur. C'est pour cette raison que Google a adopté une nouvelle approche plus robuste, s'adaptant mieux aux nouvelles données. Cette approche est appelée apprentissage fédéré(Federated learning).

L'apprentissage fédéré permet aux clients d'apprendre de manière partagée tout en conservant l'ensemble des données d'entraînement sur leurs machines et en dissociant la capacité d'apprentissage automatique de la nécessité de stocker les données côté serveur.

Le fonctionnement global de l'apprentissage fédéré est le suivant : une machine télécharge le modèle global qui est situé au niveau du serveur et l'améliore en l'entraînant avec les données de la machine. Par la suite, la machine renvoie la mise à jour du modèle au serveur. Ainsi, le serveur effectue une agrégation avec d'autres mises à jour de clients pour améliorer le modèle global. Les clients n'ont alors plus besoin de divulguer leurs propres données personnelles et ces dernières demeurent sur leurs machines. Voir figure 1.7.

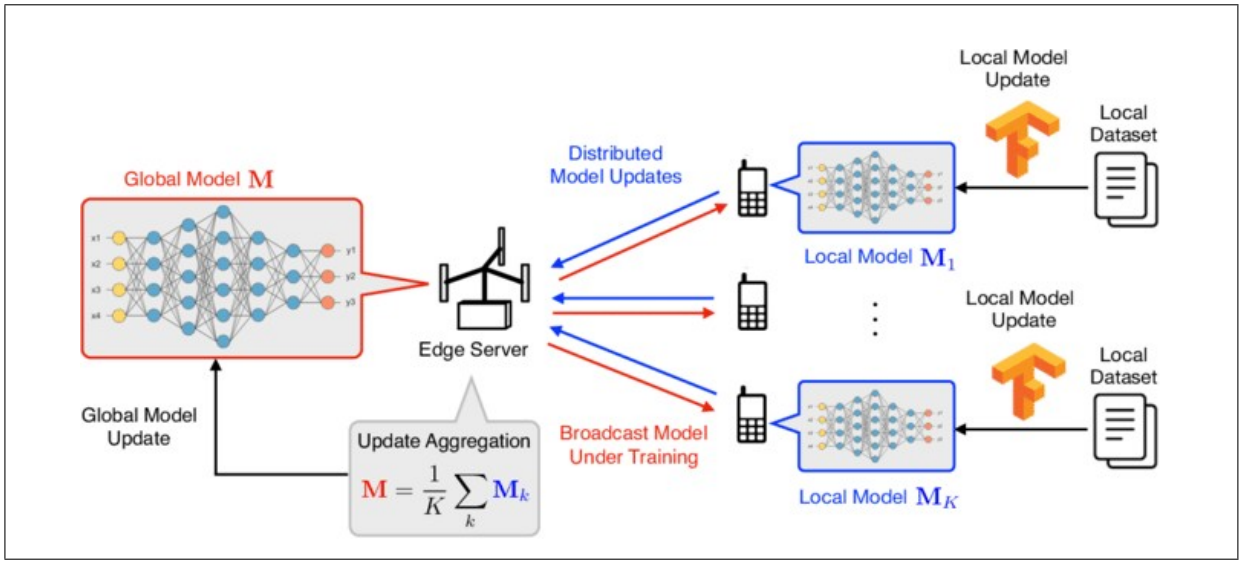

Figure 1.7 Apprentissage fédéré, tirée de Zhu (2020)

L'apprentissage fédéré est principalement composé de quatre grandes étapes :

- La première étape est le choix du serveur pour un modèle d'apprentissage.

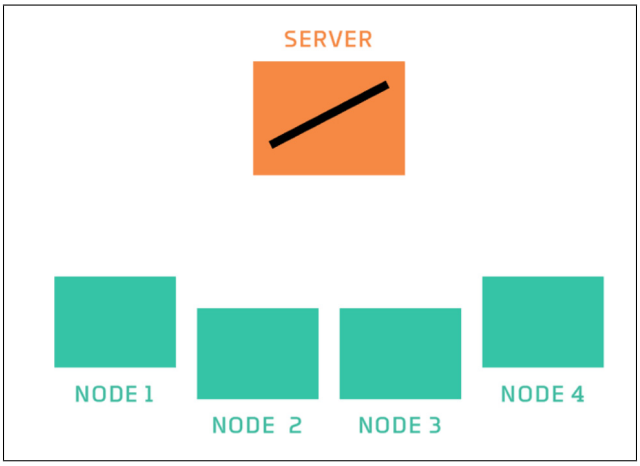

Figure 1.8 Choix du modèle, tirée de Subramaniam (2019a))

- La seconde étape consiste à effectuer la diffusion du modèle vers plusieurs périphériques décentralisés.

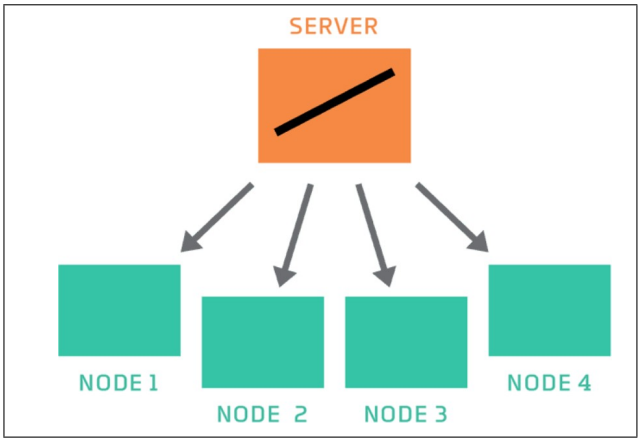

Figure 1.9 Diffusion du modèle, tirée de Subramaniam (2019a)

- La troisième étape est celle au cours de laquelle les périphériques décentralisés entament leur entraînement sur le modèle diffusé par le serveur sur leurs propres données.

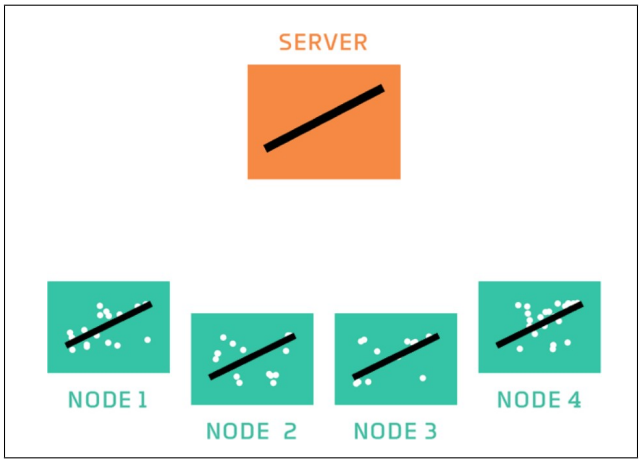

Figure 1.10 Entraînement du modèle par les périphériques, tirée de Subramaniam (2019b)

- La dernière étape consiste à récolter le résultat renvoyé par les différents modèles et représentant le gradient local puis à réaliser une agrégation de ces derniers pour produire un modèle plus performant.

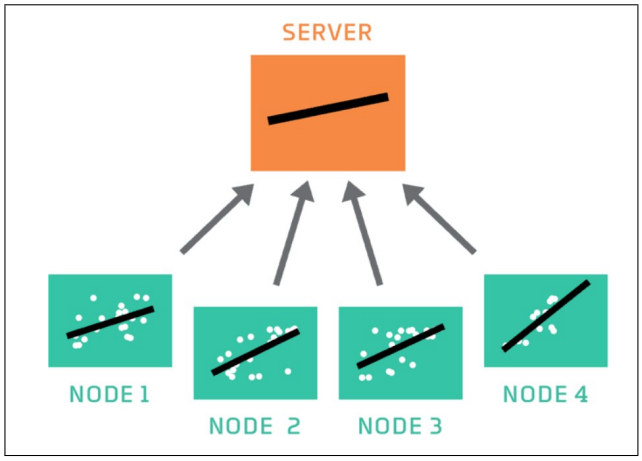

Figure 1.11 Agrégation des gradients, tirée de Subramaniam (2019c)

1.2.5.1 Sélection des clients

La sélection des clients participants se fait actuellement de manière totalement quasi aléatoire. Cependant, les résultats de l'hétérogénéité des clients et les limites de leurs ressources pourraient affecter la précision du modèle. En matière de cybersécurité, particulièrement s'agissant de détection d'intrusion, la sélection des clients dépend :

- Du nombre de clients ainsi que de l'aspect aléatoire leur choix dans chaque itération.
- Du type d'appareil utilisé par le client.
- De la puissance de calcul de l'appareil client.
- Du niveau de sécurité du réseau client.
- DiDe la disponibilité et de la fiabilité du réseau client.

1.2.5.2 Hyperparamètre de l'apprentissage fédéré

Une fois la topologie choisie, nous pouvons contrôler différents paramètres du processus d'apprentissage fédéré afin de l'optimiser par opposition à celui déjà utilisé par le modèle :

- Nombre de cycles d'apprentissage fédéré.
- Nombre total de clients utilisés dans le processus.
- Fraction d'instance client utilisée pour chaque itération.
- Taille du lot local utilisé à chaque itération d'apprentissage.

1.3 Travaux connexes

Face à la multiplication des attaques réseau, des mécanismes de détection d'intrusion ont vu le jour. Cependant, l'évolution des cyberattaques et la complexité des données réseau à traiter rendent inefficaces les méthodes de détection classiques.

Avec l'apparition du l'apprentissage automatique (ML) et le besoin immédiat de nouveaux outils de détection des menaces, le ML représente désormais un atout primordial pour contourner ces cyber-risques.

Si les méthodes classiques parviennent facilement à bloquer les attaques déjà connues, les intrusions de demain constituent toujours un défi en cybersécurité.

Dans cette section, nous discuterons de l'obtention de nos données et présenterons un récapitulatif de certains systèmes de détection d'intrusions.

1.3.1 Obtention des données

L'apprentissage automatique est utilisé pour détecter les attaques anormales à l'aide de l'entraînement sur les données. Pour ce faire, Il faut d'abord obtenir les données.

Avec les anciens jeux de données comme KDD99 ou CAIDA, l'implémentation d'un modèle efficace susceptible de détecter les intrusions d'aujourd'hui est presque impossible. Pour cette raison, Sharafaldin *et al.* (2018) ont généré un nouveau jeu de données intitulé CICIDS2017, contenant tous les critères d'attaques réseaux comme DoS, DDoS, Force Brute, XSS, Injection SQL, Infiltration, le scan des Ports et le Botnet.

Ce dernier sera labellisé par deux classes : les instances normales et les instances malveillantes. Ces instances seront caractérisées par 80 attributs. La collecte du CICIDS2017 est effectuée sur une durée de 5 jours dont le premier est uniquement réservé aux instances normales et les autres jours aux différentes attaques mentionnées au-dessus. Le CICIDS2017 est collecté à partir de deux réseaux, le premier étant le réseau victime contenant un ensemble composé de switch, routeur, pare-feu et d'un agent chargé de simuler le comportement normal pour chaque ordinateur. Le second réseau est celui des attaquants et comprend des ordinateurs avec des adresses IPs publiques et des opérations pour effectuer les attaques.

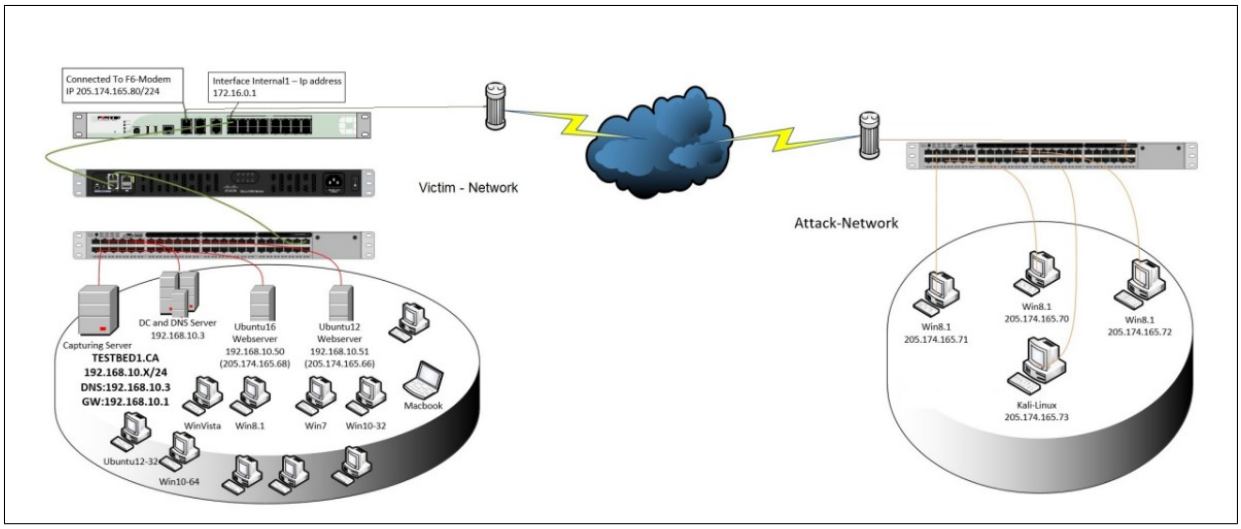

Figure 1.12 Architecture réseau du CICIDS2017, tirée de Sharafaldin *et al.* (2018)

La limite de ce jeu de données s'explique par le fait qu'il est composé de seulement deux réseaux. Cependant, dans la plupart des cas où des entreprises ont subi des attaques, la menace provenait souvent d'attaquants opérant à travers de multiples réseaux.

1.3.2 Détection d'intrusion avec l'apprentissage automatique

Après obtention des jeux de données, arrive la phase d'entrainement. Dans cette section, nous discuterons des articles et thèses qui abordent la phase de détection d'intrusion. Avant d'entamer la phase de la détection d'intrusion avec l'apprentissage fédéré il faut connaître les étapes avec l'apprentissage automatique en effet dans la thèse de Anael Beaugnon elle consiste à produire une méthodologie, un pipeline de l'apprentissage automatique, soit l'annotation des données, l'extraction des attributs, l'entraînement et l'évaluation du modèle sans introduire le jargon de ce dernier aux experts et leur permettre de se focaliser uniquement sur leurs domaines d'expertise. Le but est de guider les experts de sécurité à travers toutes ces phases afin d'apporter un système de détection d'anomalies correspondant à leurs besoins, sans pour autant connaître le côté complexe de l'apprentissage automatique à partir des solutions présentées ILAB et DIADEM.

Cependant, pour la technique de détection d'attaque, l'auteur ne mentionne pas le temps nécessaire au système pour faire un retour aux administrateurs lors d'une nouvelle attaque et les répercussions peuvent s'avérer catastrophiques sur les informations personnelles des clients. Pour cela, le système devra sauvegarder des images des modèles précédents afin d'y revenir en cas de problème.
Concernant les jeux de données, l'auteur a choisi trois jeux de données publiques, récoltées à partir de simulation ou du moteur de recherche Google, où il a entrainé ses modèles. Or, en réalité, les utilisateurs ne veulent pas partager leurs données personnelles et cela rend complexe, voire impossible, l'obtention de leurs données.c'est pour cela que l'apprentisage fédéré est primordiale.

Une autre approche a été réalisé par Haghighat, Foroushani & Li (2019) qui se fonde sur un système de détection d'intrusion SAWANT qui utilise une architecture d'apprentissage en profondeur pour analyser les données du réseau. Le jeu de données utilisé pour cela, appelé CTU 13, est récolté à partir de l'université, Czech Repulic en 2011. Il représente la combinaison de 13 jours de traffic botnet incluant également 20 millions d'instances de différents types comme IRC, P2P, http, Fast Flux, Spam, Click Fraud, Port Scan et DDos traffic. L'article extrait de ces

données plusieurs attributs significatifs en se basant sur une technique de fenêtre coulissante. Ces derniers extraits sont ensuite transférés vers une structure d'apprentissage en profondeur ANN, pour entamer l'étape de classification.

Toutefois, un problème apparait avec le modèle choisi. En effet, en cas de nouvelle attaque, ce dernier ne pourra pas la détecter facilement ou effectuer de fausses alertes, car la maintenance et la mise à jour du modèle ne peuvent être réalisées progressivement. Il rencontre également le même problème évoqué par mon mémoire précédente et relatif à la difficulté d'obtention des données d'utilisateurs impactant les performances du modèle qui ne pourra pas identifier les nouvelles attaques.

1.3.3 Détection d'intrusion avec l'apprentissage fédéré

Dans le but de préserver les données sensibles des clients, l'apprentissage fédéré est devenu indispensable. Dans cette section, nous analyserons quelques travaux appliquant l'apprentissage fédéré.

L'objectif de cet article Alazab, Priya, M, Reddy, Gadekallu & Pham (2021) est de présenter un examen approfondi de l'apprentissage fédéré dans un système de détection d'intrusion. Il s'agit d'une analyse des défis et des solutions observés dans la mise en œuvre de l'apprentissage fédéré sur divers types de systèmes de détection d'intrusion avec des approches d'apprentissage automatique pertinentes. Concernant les défis observés, l'article montre la première limitation de l'apprentissage fédéré imposée par les frais de communications par itération dans la phase d'entraînement. Un second défi est représenté par l'apparition de nouvelles attaques appelées attaques d'empoisonnement fédéré. Cette attaque se déroule dans la partie client où les étiquettes des données peuvent être facilement modifiées. Si un client prétend être un participant normal, il est ainsi capable de modifier le modèle globale pour produire des prédictions erronées. Enfin, l'apprentissage fédéré peut rencontrer des difficultés dans la gestion des ressources des appareils. En effet, certains appareils peuvent ne pas être pourvus des ressources nécessaires pour assurer l'entraînement de leurs modèles locaux. Plusieurs appareils pourraient tomber en panne, ce

qui risquerait d'entraîner un blocage sur le serveur. Des algorithmes robustes et économes en énergie doivent être mis en œuvre dans les appareils de faible puissance.

Dans cet article Chatterjee & Hanawal (2021), les auteurs ont initialement présenté une architecture pour IDS basée sur un modèle d'ensemble hybride, nommé PHEC. Ils ont ensuite adopté ce dernier à un cadre d'apprentissage fédéré. Par la suite, l'article propose une solution tolérante au bruit dans des environnements centralisés et fédérés afin d'insérer le problème du bruit dans l'étiquette. L'idée proposée utilise des classificateurs employant des fonctions de perte de substitution convexes pondérées. Le classificateur KNN est utilisé comme modèle puisque ce dernier fait preuve de robustesse contre le bruit. Enfin, les résultats expérimentaux obtenus sur différents jeux de données de référence, tirées suite à diverses attaques de sécurité, montrent que leur modèle atteint une précision de 92%sur le jeu de données NSL-KDD avec des performances proches de celles de l'apprentissage centralisé.

L'article Campos, Saura, González-Vidal, Hernández-Ramos, Bernabe, Baldini & Skarmeta (2021) évalue l'approche de l'apprentissage fédéré basé sur un classificateur multiclasse pour la détection de trois types d'attaques. Les auteurs utilisent particulièrement trois paramètres différents obtenus en partitionnant le jeu de données ToN IoT en fonction de l'adresse IP des appareils IoT et des types d'attaques. Les auteurs ont ainsi conçu trois scénarios de distribution. Le premier étant un scénario basique où chaque appareil dispose des données correspondant au trafic associé à une seule adresse IP. Le deuxième, une distribution équilibrée entre les clients, pour que chaque client dispose du même nombre d'échantillons de chaque classe. Cependant, chaque client FL peut avoir accès aux échantillons d'autres nœuds, ce qui peut entraîner des problèmes de confidentialité. Enfin, le dernier scénario utilise une distribution mixte. Ce dernier est généré dans le but d'atteindre un compromis entre les deux distributions précédentes dans lesquelles chaque partie conserve ses propres échantillons, mais ils sont équilibrés localement. Comme implémentation, IBMFL sera choisi pour l'apprentissage fédéré. L'article rapporte ainsi des expérimentations effectuées avec les trois distributions et ayant pu atteindre la meilleure précision (91%) avec le scénario équilibré. Pour le scénario mixte, le modèle a atteint une

précision de 89%. Enfin, le dernier scénario basique a enregistré la précision la plus faible avec 87%.

Les auteurs Sarhan, Layeghy, Moustafa & Portmann (2021) divulguent les performances de l'apprentissage fédéré sur différents jeux de données et modèles. En effet, l'article "A Cyber Threat Intelligence Sharing Scheme basedon Federated Learning for Network Intrusion Detection" , ayant choisi comme modèles le DNN et le LSTM, a assuré l'entraînement de ces derniers sur les deux jeux de données NF-UNSW-NB15-v2 et NF-BoT-IoT-v2. Pour chaque jeu de données, une classification binaire et multiclasse a été effectuée sur l'apprentissage centralisé et fédéré. En effet, avec le jeu de données NF-UNSW-NB15-v2, le modèle ayant enregistré la meilleure précision pour l'approche fédérée et centralisée est le DNN avec respectivement 90.83% et 99.38% pour la classification binaire, et 85.06% et 99.41% pour la classification multiclasse. Néanmoins, pour NF-BoT-IoT-v2, le LSTM offre une meilleure prédiction de quelques pourcentages sur l'ensemble des scénarios. L'article Intelligent Intrusion Detection Based on Federated Learning for Edge-Assisted Internet of Things rapporte la sélection du modèle CNN pour le jeu de données NSL-KDD ayant atteint une précision de 94.67%.

1.3.4 Lacune des travaux existants

Tous ces travaux offrent des solutions diverses pouvant détecter les intrusions réseau. Cependant, on peut constater que la majorité des articles se focalisent sur des jeux de données équilibrées et avec seulement une sélection de clients aléatoires. On a remarqué qu'aucun de ces travaux n'apporte une solution pour des jeux de données non équilibré comme IDS2017 et IDS2018 et n'apporte pas d'autre stratégie de sélection des clients afin d'optimiser l'entraînement en matière de temps d'exécution et amélioration importante de la précision en moins de cycles d'entraînement.

1.4 Conclusion

Dans ce premier chapitre, nous avons expliqué la procédure actuelle suivie d'une présentation de la vue globale sur l'environnement du projet.
Nous avons mentionné les travaux connexes ainsi que le nouveau jeu de donnée IDS2017.
Dans le chapitre suivant, nous présentons la solution proposée à la problématique.

CHAPITRE 2

SOLUTION PROPOSÉE

2.1 Introduction

Dans ce chapitre, nous proposerons et évaluerons une méthode d'apprentissage fédéré. Par la suite, nous présenterons les jeux de données basés sur deux réseaux composés de nœuds victimes et attaquants, IDS2017 et IDS2018, contenants chacun une multitude d'attaques.

Dans les sections suivantes, nous présenterons l'approche d'apprentissage fédéré et la méthode de préparation des données pour un usage dans un cadre expérimental.

2.2 L'approche de l'apprentissage fédéré

Afin de respecter la vie privée des utilisateurs et d'offrir un niveau de détection des intrusions réseau plus efficaces et qui reste à jour vu que l'entraînement va se faire sur les modèles des machines des utilisateurs et vont partager leurs résultats avec le modèle global. On a ainsi opté pour l'apprentissage fédéré. Cette approche est divisée en deux parties, la partie serveur contenant le modèle global et la partie client représentant les autres périphériques (ordinateur, laptop,...). Le rôle du serveur, à chaque itération, est de partager et de mettre à jour le modèle. Ce modèle est appelé le modèle global. Le côté client est composé de plusieurs machines représentant les éventuelles victimes visées par les cyberattaques. Chaque machine possède ses propres données constituées d'informations sur le flux du réseau (malveillantes et bénignes). Par conséquent, chaque client utilisera les données étiquetées pour entraîner un modèle local sans nécessairement recourir au partage d'informations sensibles. Le processus est le suivant : à chaque itération, les clients obtiennent une copie du modèle global et l'entraînent sur leurs données locales. Ensuite, un par un, ils enverront les poids agrégés au serveur où ils sont moyennés. C'est alors au tour du serveur d'améliorer le modèle global en apprenant de nouveaux modèles puis de distribuer le nouveau modèle aux clients. Au fil du temps, le modèle devient progressivement plus efficace.(voir figure 2.1)

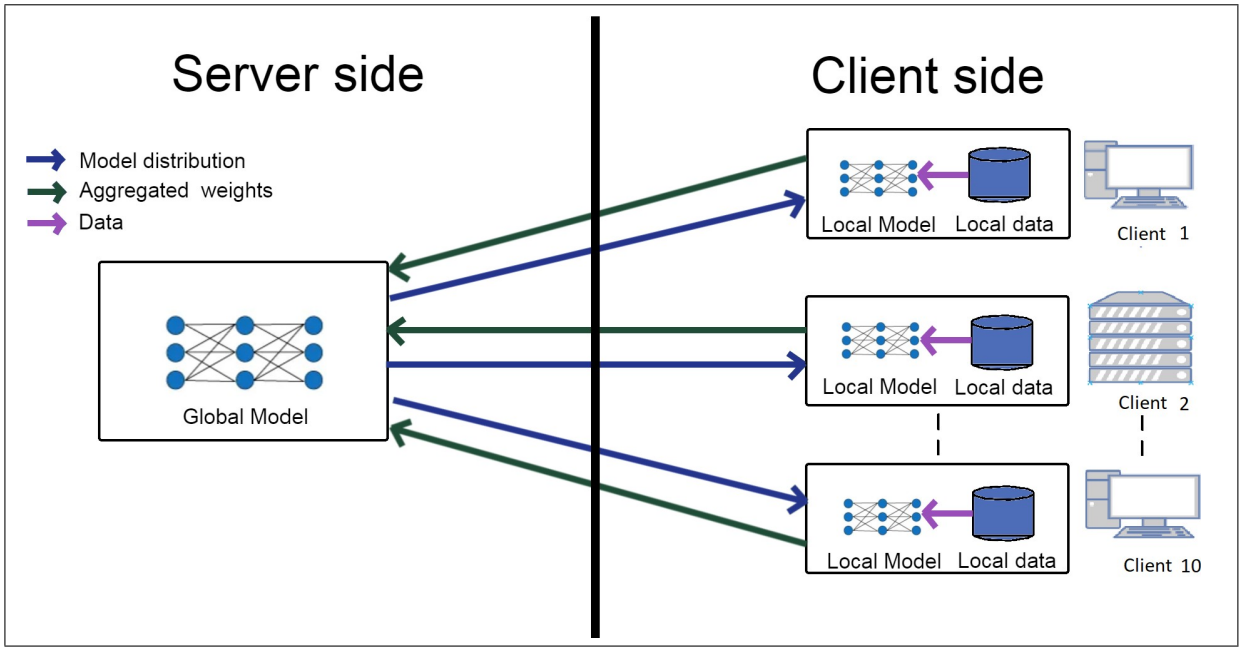

Figure 2.1 Architecture de l'apprentissage fédéré

Le but de cette approche est de prédire si l'instance est normale ou ou si une attaque est imminente et de rester à jour.

Pour évaluer l'algorithme d'apprentissage fédéré, nous avons utilisé un ensemble de données intitulé IDS2017, situé côté serveur et dont les données n'ont pas encore été entraînées par le modèle.

2.3 Création d'un ensemble de données pour la détection d'intrusion réseau basée sur l'apprentissage fédéré

Expérimenter une approche FL et étudier ses performances représentent un véritable challenge! En fait, l'approche courante dans la communauté FL consiste à utiliser un ensemble de données ML à la pointe de la technologie et à simuler sa distribution sur un ensemble de clients. Dans une telle configuration d'expérimentation, les chercheurs ont l'habitude de faire varier le nombre de clients et d'étudier diverses stratégies de distribution d'échantillons sur eux. Dans le cadre de la détection d'intrusions réseau, les scénarios d'expérimentation doivent être limités par la topologie du réseau, les caractéristiques du nœud, etc. En effet, il n'est pas réaliste d'attribuer

un paquet (échantillon), qui est échangé entre un nœud source A et un nœud destination B, à n'importe quelle machine du réseau (C par exemple) ! De plus, le nombre de clients FL possibles est limité par le nombre de nœuds réels appartenant au réseau protégé. Ainsi, la topologie du réseau et l'adresse IP réelle de chaque nœud du réseau sont requises avant toute distribution de Dataset sur un ensemble d'apprentissages des clients. Cela permettra une distribution réaliste de l'ensemble des données d'origine sur les nœuds du réseau.

Afin de préparer l'ensemble des données pour notre expérimentation, nous avons besoin de la disponibilité des informations de topologie du réseau et des adresses IP source et de destination dans chaque échantillon de paquet de l'ensemble de données. Sur la base de ces informations, nous reconnaissons chaque adresse IP dans la topologie du réseau en tant que client d'apprentissage possible. La distribution des échantillons de paquets de l'ensemble de données d'origine sur les clients d'apprentissage s'effectuera de la manière suivante :

- Si l'adresse IP de destination appartient au réseau, alors une copie de ce paquet appartiendra à l'ensemble des données « locales » du client identifié par cette adresse IP. En d'autres termes, étant donné qu'un nœud de réseau peut « voir » chaque paquet reçu, il l'utilisera pour entraîner son modèle local.
- Si l'adresse IP source appartient au réseau, alors une copie de ce paquet appartiendra à l'ensemble des données « locales » du client identifié par cette adresse IP. En d'autres termes, étant donné qu'un nœud de réseau peut « voir » chaque paquet envoyé, il l'utilisera pour entraîner son modèle local.

Sur la base de l'approche de distribution d'échantillons susmentionnée, lorsqu'il s'agit d'un broadcast, un paquet peut apparaître dans plusieurs nœuds du réseau.

2.4 Ensemble de données : IDS 2017

L'ensemble des données utilisé pour valider l'expérimentation IDS2017 a été publié en 2017 par l'Institut canadien pour la cybersécurité et intitulé « IDS2017 » Haghighat, Foroushani & Li

(2019). La topologie du réseau est composée de deux réseaux (Fig. 2.2). Le premier réseau est le réseau victime qui est composé de 6 ordinateurs de bureau, 1 ordinateur portable et 3 serveurs avec différents systèmes d'exploitation installés et les adresses IP publiques et privées associées. Le second réseau est le réseau d'attaque qui est composé de 4 postes de travail avec Kali Linux ou Windows 8.1 comme systèmes d'exploitation. Chaque réseau est connecté à Internet via un équipement commutateur-routeur.

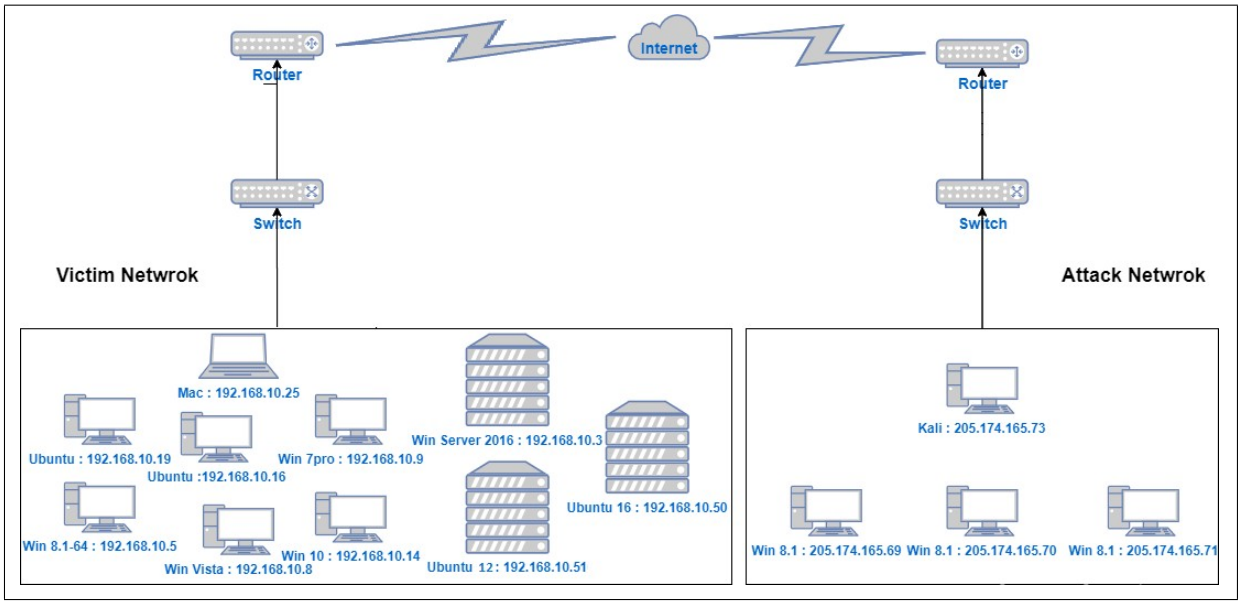

Figure 2.2 Réseau IDS 2017, tirée de Sharafaldin *et al.* (2018)

Le jeu de données est constitué d'un trafic réseau collecté pendant 5 jours : du 3 juillet au 7 juillet 2017. Les classes de trafic collectées chaque jour sont répertoriées dans le Tableau 2.1. Le trafic peut être normal ou malveillant.

Tableau 2.1 Étiquette des données par jour

Jour	Label
Lundi	Normal
Mardi	BForce/SFTP/SSH
Mercredi	Web and Infiltration Attacks Web BForce, XSS and Sql Inject. Infiltration Dropbox Download and Cool disk
Jeudi	Web and Infiltration Attacks Web BForce, XSS and Sql Inject. Infiltration Dropbox Download and Cool disk
Vendredi	DDoS LOIT, Botnet ARES, PortScans (sS,sT,sF,sX,sN,sP,sV,sU, sO,sA,sW,sR,sL and B)

Les cyberattaques appartenant au trafic malveillant IDS2017 sont décrites ci-dessous :

- Attaque Bruteforce : l'attaquant est une machine Kali Linux ayant l'adresse IP 205.174.165.73 et la victime est le serveur web exécutant un système d'exploitation Ubuntu 16.04 et ayant l'adresse IP 192.168.10.50.
- Attaque DoS : l'attaquant est une machine Kali Linux ayant l'adresse IP 205.174.165.73 et la victime est une machine Ubuntu 12.04 ayant l'adresse IP 192.168.10.51.
- Attaque Web : l'attaquant est une machine Kali Linux ayant l'adresse IP 205.174.165.73 et la victime est un serveur Web exécutant un système d'exploitation Ubuntu 16.04 et ayant l'adresse IP 192.168.10.50.
- Attaque par infiltration : l'attaquant est une machine Kali Linux ayant l'adresse IP 205.174.165.73 et les victimes sont diverses machines exécutant différents systèmes d'exploitation : Windows, Ubuntu et Macintosh.

- Attaque par botnet : L'attaquant est une machine Kali Linux ayant l'adresse IP 205.174.165.73 et les victimes sont des machines Windows : exécutant Windows Vista (192.168.10.8), Windows 7 (192.168.10.9), Windows 8.1 (192.168.10.5) ou Windows 10 (192.168.10.14).
- Attaque DDoS et PortScan : les attaquants sont les 3 postes de travail du réseau d'attaque exécutant les systèmes d'exploitation Windows 8.1 et ayant les adresses IP 205.174.165.69, 205.174.165.70 et 205.174.165.71. La machine victime est le serveur Web exécutant le système d'exploitation Ubuntu 16 et ayant l'adresse IP 192.168.10.50 ainsi que toutes les machines Windows : adresses IP 192.168.10.8, 192.168.10.9, 192.168.10.5 et 192.168.10.14.

Le nombre d'échantillons normaux et d'attaques (paquets) est indiqué dans le tableau 2.2.

Tableau 2.2 Dataset IDS2017 - nombre de paquets par attaque

Instance	**Total**
BENIGN	2104911
Bot	1966
DDoS	128027
DoS GoldenEye	10293
DoS Hulk	231073
DoS Slowhttptest	5499
DoS slowloris	5796
FTP-Patator	7938
Heartbleed	11
Infiltration	36
PortScan	158930
SSH-Patator	5897

L'ensemble de données contient des enregistrements étiquetés « bénins » et plusieurs types d'attaques comme Brute Force FTP, Brute Force SSH, DOS, Heartbleed, Web Attack, Infiltration, Botnet et DDoS. Nous avons regroupé ces attaques dans une classe intitulée « attaque » pour effectuer une classification binaire (normale vs anormale). En appliquant l'approche détaillée à la sous-section III. B, nous avons identifié chaque machine du réseau en tant que client et attribué à chaque client un sous-ensemble des échantillons du jeu de données IDS2017. Le nombre d'échantillons attribués à chaque client est présenté dans le tableau 2.3.

Tableau 2.3 Distribution des instances de l'IDS2017 sur les Clients FL

Client	**Adresse IP**	**Étiquette**	**Total ligne**
0	192.168.10.3	normale	905365
0	192.168.10.3	attaque	0
1	192.168.10.50	normal	105395
1	192.168.10.50	attaque	552373
2	192.168.10.51	normal	96360
2	192.168.10.51	attaque	11
3	192.168.10.19	normal	188693
3	192.168.10.19	attaque	0
4	192.168.10.16	normal	320553
4	192.168.10.16	attaque	0
5	192.168.10.9	normal	177716
5	192.168.10.9	attaque	372
6	192.168.10.5	normal	211680
6	192.168.10.5	attaque	287
7	192.168.10.8	normal	236163
7	192.168.10.8	attaque	407
8	192.168.10.14	normal	327407
8	192.168.10.14	attaque	922

Client	Adresse IP	Étiquette	Total ligne
9	192.168.10.25	normal	189928
9	192.168.10.25	attaque	0

Nous avons équilibré l'ensemble de données. L'ensemble des données résultant a le même nombre d'échantillons normaux et d'attaques (554374). De cette façon, le modèle formé sera capable d'apprendre le modèle de chaque classe de manière égale.

2.5 Ensemble de données : IDS 2018

L'ensemble de données IDS2018 a été publié en 2018 parl'Institut canadien pour la cybersécurité et comprend sept scénarios d'attaque différents : Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration.

Le réseau est composé de 5 départements pour un total de 420 machines et 30 serveurs victimes et les attaquants sont composés de 50 machines qui vont appliquer de multiples attaques sur une durée de 11 jours, comme illustré dans la figure 2.3.

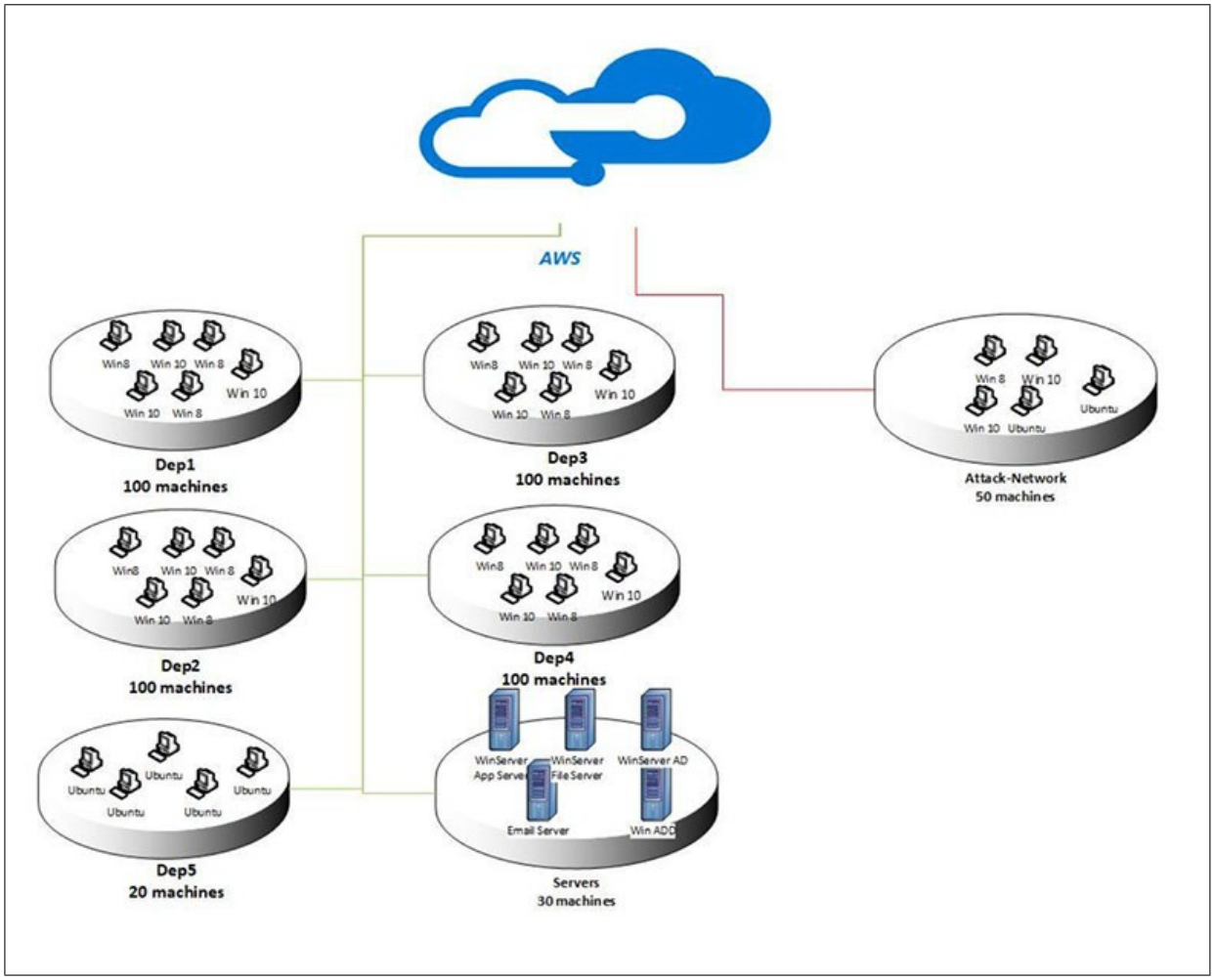

Figure 2.3 Réseau IDS 2018, tirée de of new brunswick (2018)

Le tableau (Table 2.4) illustre l'adresse ip de l'attaquant, le type d'attaque ainsi que la date des attaques effectuées.

- Mercredi 14-02-2018 FTP et SSH Bruteforce.
- Jeudi-15-02-2018 DoS-GoldenEye et DoS-Slowloris
- Vendredi-16-02-2018 DoS-SlowHTTPTest et DoS-Hulk
- Mardi-20-02-2018 DDoS attacks-LOIC-HTTP et DDoS-LOIC-UDP
- Mercredi-21-02-2018 DDoS-LOIC-UDP et DDoS-HOIC

- Jeudi-22-02-2018 Brute Force Web, XSS et SQL Injection
- Vendredi-23-02-2018 Brute Force Web et Brute Force -XSS
- Mercredi-28-02-2018 Infiltration
- Jeudi-01-03-2018 Infiltration
- Vendredi-02-03-2018 Bot

Tableau 2.4 Caractéristiques de la machine

Attaquant	**Type**	**Jour**
172.31.70.4	FTP-BruteForce	Mercredi-14-02-2018
172.31.70.6	SSH-Bruteforce	Mercredi-14-02-2018
172.31.70.46	DoS-GoldenEye	Jeudi-15-02-2018
172.31.70.8	DoS-Slowloris	Jeudi-15-02-2018
172.31.70.23	DoS-SlowHTTPTest	Vendredi-16-02-2018
172.31.70.16	DoS-Hulk	Vendredi-16-02-2018
18.218.115.60 / 18.219.9.1 18.219.32.43 / 18.218.55.126 52.14.136.135 / 18.219.5.43 18.216.200.189 / 18.218.229.235 18.218.11.51 / 18.216.24.42	DDoS attacks-LOIC-HTTP	Mardi-20-02-2018
18.218.115.60 / 18.219.9.1 18.219.32.43 / 18.218.55.126 52.14.136.135 / 18.219.5.43 18.216.200.189 / 18.218.229.235 18.218.11.51 / 18.216.24.42	DDoS-LOIC-UDP	Mardi-20-02-2018

Attaquant	**Type**	**Jour**
18.218.115.60 / 18.219.9.1 18.219.32.43 / 18.218.55.126 52.14.136.135 / 18.219.5.43 18.216.200.189 / 18.218.229.235 18.218.11.51 / 18.216.24.42	DDoS-LOIC-UDP	Mercredi-21-02-2018
18.218.115.60 / 18.219.9.1 18.219.32.43 / 18.218.55.126 52.14.136.135 / 18.219.5.43 18.216.200.189 / 18.218.229.235 18.218.11.51 / 18.216.24.42	DDoS-HOIC	Mercredi-21-02-2018
18.218.115.60	Brute Force Web	Jeudi-22-02-2018
18.218.115.60	Brute Force -XSS	Jeudi-22-02-2018
18.218.115.60	SQL Injection	Jeudi-22-02-2018
18.218.115.60	Brute Force Web	Vendredi-23-02-2018
18.218.115.60	Brute Force -XSS	Vendredi-23-02-2018
13.58.225.34	Infiltration	Mercredi-28-02-2018
13.58.225.34	Infiltration	Jeudi-01-03-2018
18.219.211.138	Bot	Vendredi-02-03-2018

Afin d'exposer une présentation claire, nous avons choisi de mettre en évidence la distribution des instances pour dix clients uniquement (Tableau 2.5) :

Tableau 2.5 Distribution des instances de l'IDS2018 sur les Clients FL

Client	**Étiquette**	**Total ligne**
0	Normale	184854
1	Normale	20676
1	DDos	128027
1	Dos GoldenEye	10293
Client	**Étiquette**	**Total ligne**
1	Dos Hulk	231073
1	Dos Slowhttptest	5499
1	Dos slowloris	5796
1	FTP-Patator	7938
1	PortScan	158930
1	SSH-Patator	5897
1	Web Attack Brute Force	1507
1	Web Attack Sql Injection	21
1	Web Attack XSS	652
2	normale	19318
2	Heartbleed	11
3	Normale	39201
4	Normale	64062
4	Bot	2
5	Normale	35683
5	Bot	372
6	Normale	42405
6	Bot	288
7	Normale	46406
7	Bot	374

7	Infiltration	36
8	Normale	68404
8	Bot	928
9	Normale	36635

2.6 Conclusion

Nous avons consacré ce chapitre à la solution proposée. Nous avons initialement expliqué l'approche de l'apprentissage fédéré, puis la démarche adoptée pour la création d'un ensemble de données adéquates à l'apprentissage fédéré et enfin nous nous sommes focalisés sur l'étape la plus primordiale qui est celle du pré-traitement des données. Dans cette dernière, nous avons procédé à la combinaison de plusieurs ensembles de données, puis à l'encodage one-hot des labels des instances normales et attaques et avons finalement mis en échelle les données d'entraînement et de test.

CHAPITRE 3

EXPÉRIMENTATION ET ÉVALUATION

3.1 Introduction

Au cours de ce chapitre qui constitue le dernier volet du rapport, nous présenterons la partie expérimentale de notre projet ainsi que son évaluation. L'expérimentation détaille le jeu de données et présente nos métriques d'évaluation des résultats lors de l'expérimentation. Nous finirons par la présentation des résultats de nos expérimentations tout en les expliquant.

3.2 Expérimentation

Dans cette section, nous entamons la répartition de notre jeu de données et la présentation de nos choix comme métrique d'évaluation.

3.2.1 Le fractionnement du jeu de données

Au cours du chapitre1, nous avons défini la répartition du jeu de données, en ce qui concerne l'apprentissage automatique, en éclaircissant leurs rôles et le besoin de cette répartition. Avant de commencer à expliquer les expérimentations faites au cours de ce projet, nous présentons donc cette répartition. (voir figure 3.1)

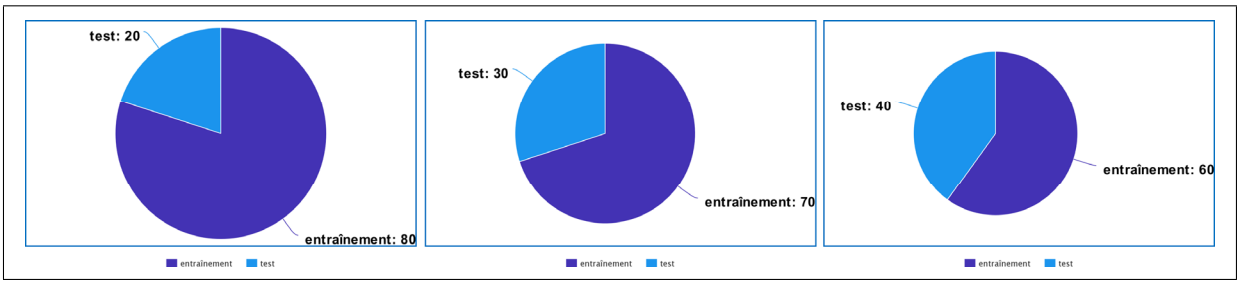

Figure 3.1 Répartition du jeu de données

En disposant d'un ensemble d'apprentissages plus important, nous donnons à notre algorithme une meilleure chance de comprendre les données. Ainsi, il pourra mieux faire la distinction entre une instance normale et une attaque.

3.2.2 Métrique : la perte

Nous avons calculé la perte pour notre modèle, en utilisant la fonction de perte "L'entropie croisée". Cette dernière compare la prédiction du modèle à l'étiquette qui est la vraie distribution de probabilité. C'est une mesure de performance dont la sortie est une valeur de probabilité comprise entre 0 et 1.

La perte d'entropie croisée augmente à mesure que la probabilité prédite diverge de l'étiquette réelle et, vice-versa, la perte d'entropie croisée diminue à mesure que la prédiction devient de plus en plus précise. Réduire la valeur de la perte de l'entropie représente l'objectif à atteindre pour assurer une meilleure précision La formule est la suivante :

$$H(p, q) = \mathrm{E}_p[-\log q] \tag{3.1}$$

3.2.3 Métrique : la précision

La métrique de précision évalue la précision de la prédiction du modèle par rapport aux données réelles. Seulement cette métrique a été utilisé pour l'apprentissage fédéré puisque les méthodes d'évaluation des modèles fédérés ne sont pas encore développées par l'outil Tensorflow.

$$Pre'cision = \frac{nombre\ \ des\ \ correctes\ \ \ \ pre'diction}{nombre\ \ total\ \ de'dictions\ \ f\ aites} \tag{3.2}$$

3.2.4 Métrique : sensibilité et spécificité

Une autre métrique qui permet d'évaluer les performances d'un modèle est la sensibilité et spécificité.

- VP (vrais positifs) représente le nombre d'instance d'attaque avec une prédiction d'attaque.
- FP (faux positifs) représente le nombre d'instance normale avec une prédiction d'attaque.
- FN (faux négatifs) représente le nombre d'instance d'attaque avec une prédiction normale.
- VN (vrais négatifs) représente le nombre d'instance normale avec une prédiction normal.

La sensibilité, ou la probabilité que le modèle prédise une attaque lorsqu'il s'agit d'une instance malveillante, se mesure comme suit : VP/VP+FN Une mesure de la sensibilité s'accompagne toujours d'une mesure de la spécificité. Ainsi, la spécificité, ou la probabilité d'obtenir une prédiction normale sur une instance non dangereuse, est obtenue par : VP/VP+FP Afin de mieux comprendre cette métrique, voir la figure 3.2 où les points représente les instances. les éléments qui se trouvent à l'intérieur du cercle sont les instances détecter par le modèle. Pour la partie avec la couleur rouge. elle représente les instances d'attaques et pour la couleur verte, elle représente les instances normales.

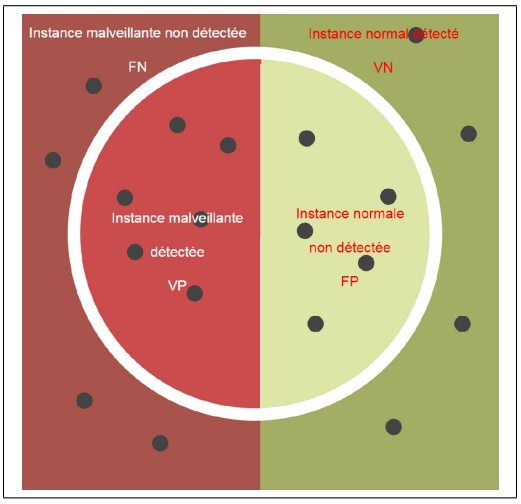

Figure 3.2 Sensibilité et spécificité, tirée de Datamok (2021)

3.3 Expérimentation : IDS 2017

Nous étudions l'efficacité de l'algorithme fédéré en utilisant un modèle CNN et une classification binaire. Le modèle comporte 74 couches d'entrée et 1 couche de sortie correspondant respectivement au nombre d'attributs et au nombre de classes. Les couches de sortie sont équipées de softmax comme fonction d'activation. Le modèle comporte 3 couches convolutives et 3 couches entièrement connectées. Les neurones sont équipés de la fonction d'activation Relu. Considérant que nous avons une sortie binaire, nous avons choisi de travailler avec une entropie croisée binaire comme fonction de perte.

Nous utilisons un filtre de taille 3 et un taux d'apprentissage de 0,5, des tailles de lots de 32 et 64, 7 époques et 8 machines par tour. Chaque époque aura 10 tours et chacun d'entre eux utilise seulement 10% du sous-ensemble de chaque appareil. À chaque expérience, nous avons modifié le pourcentage des données utilisées comme ensemble d'entraînement pour chaque appareil à 60,70 et 80% et l'ensemble de tests à 40,30 et 20 ainsi que la sélection du client à chaque tour.

Pour évaluer notre modèle, nous avons procédé à la visualisation des métriques sur 70 itérations. Nous avons eu recours à quatre stratégies de sélection des clients :

- Tous les clients sont sélectionnés
- Clients aléatoires sélectionnés
- Sélection des clients en fonction des attaques
- Sélection en fonction du système d'exploitation

3.3.1 Tous les clients sont sélectionnés

Les comportements de convergences de l'algorithme fédéré pour la stratégie où tous les clients sont sélectionnés sont montrés dans les figures 3.3, 3.4.

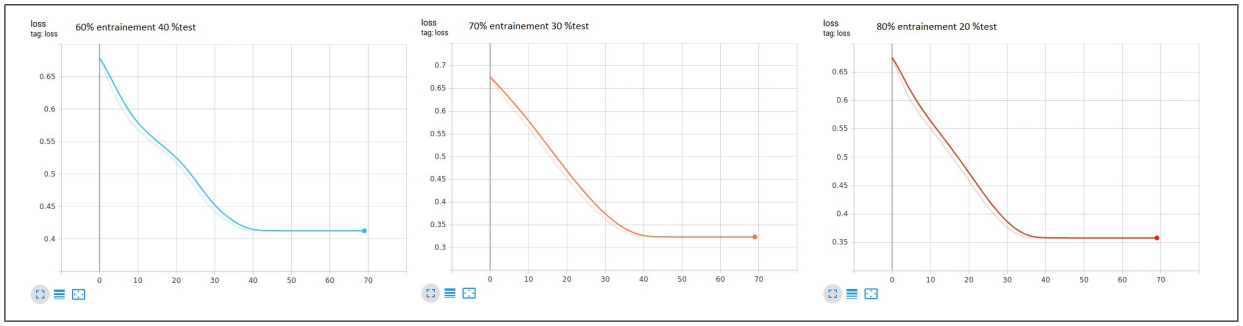

Figure 3.3 Visualisation de la perte de l'apprentissage sur tous les clients

Les courbes présentent une forme monotone lisse avec une diminution de la perte lors de l'apprentissage et de la validation sur toutes les répartitions des jeux de données 60%, 70% et 80%.

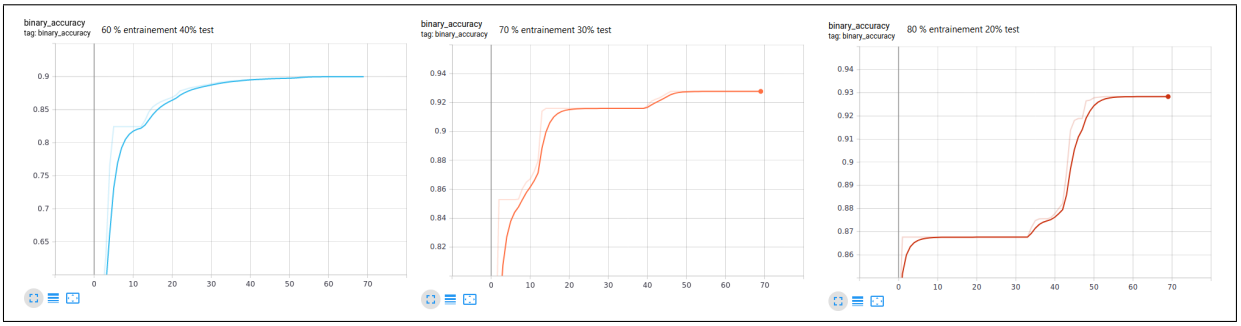

Figure 3.4 Visualisation de la précision de l'apprentissage sur tous les clients

Quant à la précision, la courbe de l'apprentissage a de grandes variations au début mais a atteint une convergence à la fin des itérations. Ceci conclue que notre modèle prédit correctement les étiquettes.

3.3.2 Clients aléatoires sélectionnés

Avec enjeu du temps et de ressources plus réel où certaines machines peuvent tomber en panne, nous avons choisi de sélectionner un nombre aléatoire de clients à chaque tour. Les courbes

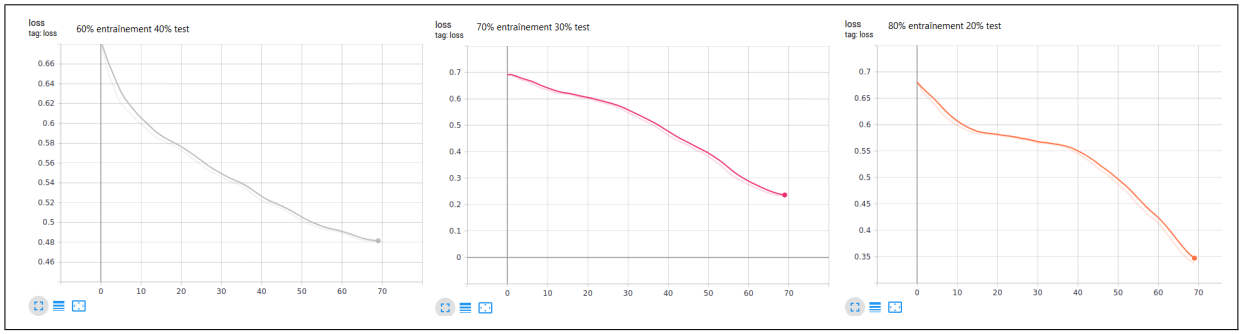

Figure 3.5 Visualisation de la perte de l'apprentissage sur des clients aléatoires

montrent une diminution de la perte, mais aucune convergence en 70 itérations. Donc, avec plus d'itérations, notre modèle pourrait s'avérer plus efficace.

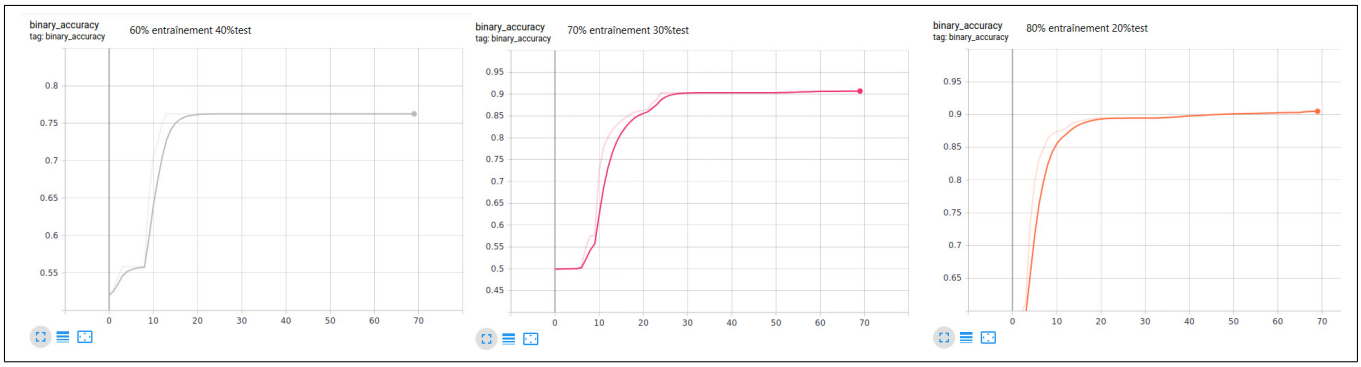

Figure 3.6 Visualisation de la précision de l'apprentissage sur des clients aléatoires

Nous avons découvert que même avec une sélection aléatoire de clients à chaque tour, le paramètre de distribution de 70% entraînement obtient la meilleure précision avec un taux de 91%.

3.3.3 Sélection des clients en fonction des attaques

Dans cette partie, la sélection du client est différente d'un tour à l'autre, selon le type d'attaque. Cela signifie que seules les instances appartenant à un type d'attaque (exemple DDos) seront sélectionnées :

- Tour 1 : focalisation sur le client attaqué par bruteforce qui est le client 1 avec l'adresse IP 192.168.10.50
- Tour 2 : entraînement sur les clients attaqués par Dos qui sont les clients 1 et 4 avec les adresses IP 192.168.10.50 et 192.168.10.16
- Tour 3 : le client sélectionné est la cible d'attaque web qui est le client 1 avec l'adresse IP 192.168.10.50
- Tour 4 : l'algorithme fédéré sélectionne les clients avec une attaque par infiltration qui sont tous des clients sauf les clients 0, 1 et 2
- Tour 5 : Sélection des clients avec attaque Botnet qui sont les clients 5, 6, 7 et 8 avec l'adresse IP 192.168.10.9, 192.168.10.5, 192.168.10.8, 192.168.10.14 .
- Tour 6 : focalisé sur le client assailli par une attaque DDos et Portscan qui est le client 1 avec l'adresse IP 192.168.10.50.
- Tour 7 : L'algorithme d'apprentissage fédéré sera entraîné en sélectionnant tous les clients.

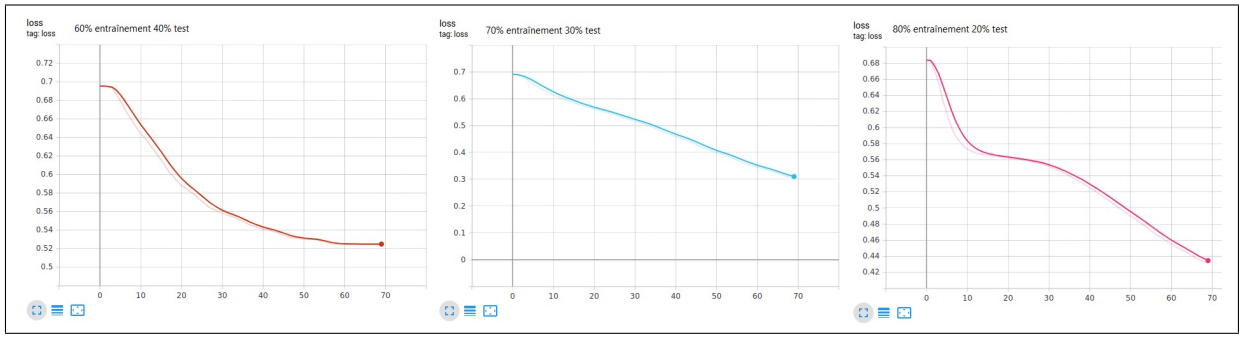

Figure 3.7 Visualisation de la perte de l'apprentissage sur les clients en fonction des attaques

Pour la distribution de 60% entraînement, nous pouvons cependant observer une convergence pour la distribution 70% et 80% avec 70 itérations, il est encore difficile d'expliquer cette observation.

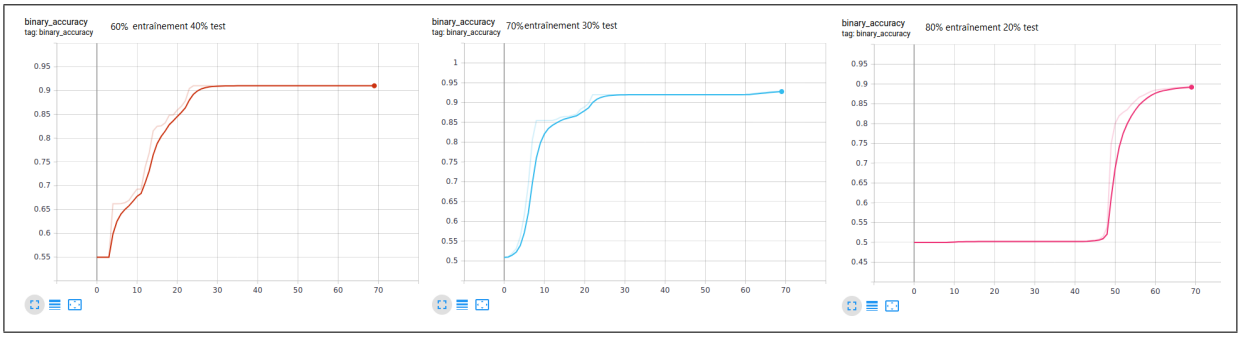

Figure 3.8 Visualisation de la précision de l'apprentissage sur les clients en fonction des attaques

Par conséquent, nous avons constaté que même la distribution de 70% est une piste prometteuse avec un meilleur taux de précision de 93%.

3.3.4 Sélection du client en fonction du système d'exploitation

Dans cette sous-section, la sélection du client se concentre sur les systèmes d'exploitation disponibles sur le réseau où, à chaque tour, nous désignons un seul type de système d'exploitation par client :

- Tour 1 : Les clients sélectionnés seront 6, 3, 1, 9 et 0 qui ont respectivement Windows 7 pro, Ubuntu, Ubuntu 12, Mac et Win server 2016 comme système d'exploitation avec les adresses IP 192.168.10.5, 192.168.10.19, 192.168.10.50, 192.168.10.25 et 192.168.10.3
- Tour 2 : Le système d'exploitation sélectionné dans ce tour sera Windows 7 pro client 5, Ubuntu client 4, Ubuntu 12 client 2, Mac client 9 et Win server 2016 client 0 avec l'adresse IP 192.168.10.19 , 192.168.10.16
- Tour 3 : Formation sur les clients 8 avec Windows 7 pro, client 3 avec Ubuntu, client 1 avec Ubuntu 12, client 9 avec Mac et client 0 avec Win server 2016 comme système d'exploitation avec les adresses IP 192.168.10.14, 192.168.10.19, 192.168.10.50, 192.168.10.25 et 192.168.10.3

- Tour 4 : L'algorithme fédéré sélectionne le client 7, 4, 2, 9 et 0 qui ont les systèmes d'exploitation Windows 7 pro, Ubuntu, Ubuntu 12, Mac et Win server 2016 avec l'adresse IP 192.168.10.8, 192.168.10.16, 192.168.10.51, 192.168.10.25 et 192.168.10.3

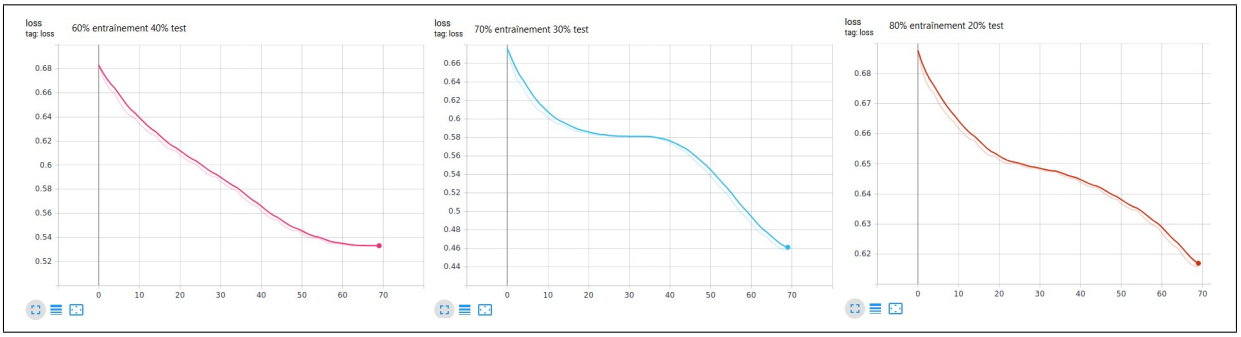

Figure 3.9 Visualisation de la perte de l'apprentissage sur les clients en fonction des systèmes d'exploitation

Nous pouvons conclure que notre modèle enregistre des améliorations au fil des itérations et qu'il n'y a pas sur-apprentissage, puisqu'une diminution de la perte est constatée tout au long des 70 itérations.

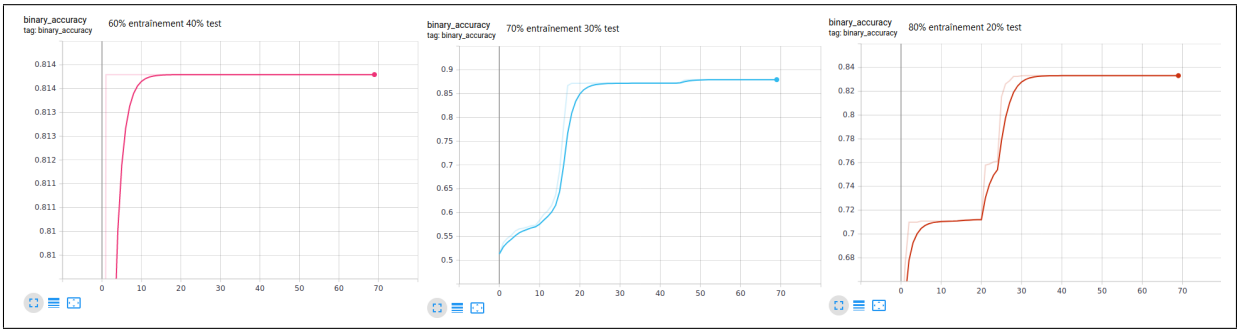

Figure 3.10 Visualisation de la précision de l'apprentissage sur les clients en fonction des systèmes d'exploitation

Nous avons déterminé qu'avec le choix des clients ayant des systèmes d'exploitation différents à chaque tour, le paramètre 70% entraînement obtient la meilleure précision avec un taux de 88%.

3.3.5 Comparaison entre les quatre approches

Après avoir expérimenté les quatre approches de sélection des clients sur différentes sections de l'ensemble de données, nous avons constaté que l'algorithme d'apprentissage fédéré est fiable même si tous les clients ne sont pas disponibles à chaque tour en raison d'une panne de courant ou d'une perte de connexion.

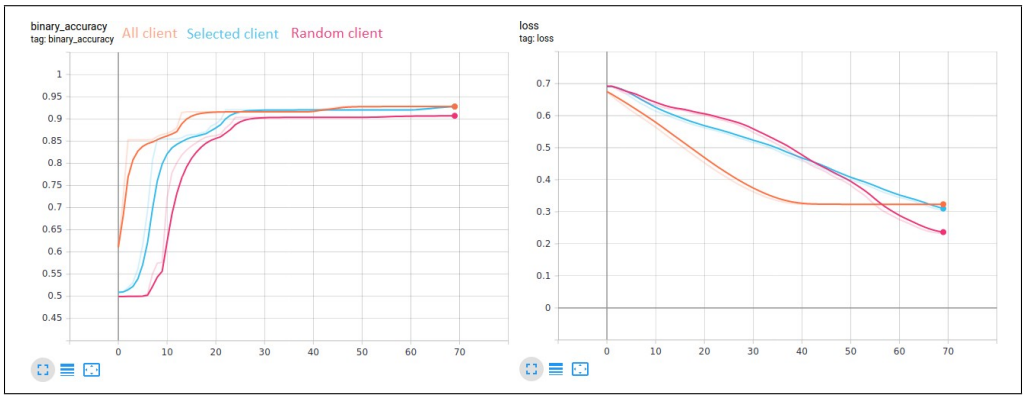

Figure 3.11 Comparaison entre le taux de précision et la perte

Comme nous pouvons le voir sur la figure 3.11, la précision de l'algorithme fédéré avec sélection de clients en fonction du type d'attaque est la même que celle de tous les clients sélectionnés à chaque tour.

Dans la figure 3.12, le résultat de la précision de l'apprentissage fédéré est inférieur à la précision de l'apprentissage dans une méthode centralisée. Cependant, il offre plus de confidentialité aux clients, car il ne nécessite pas leurs informations sensibles en tant que données de formation.

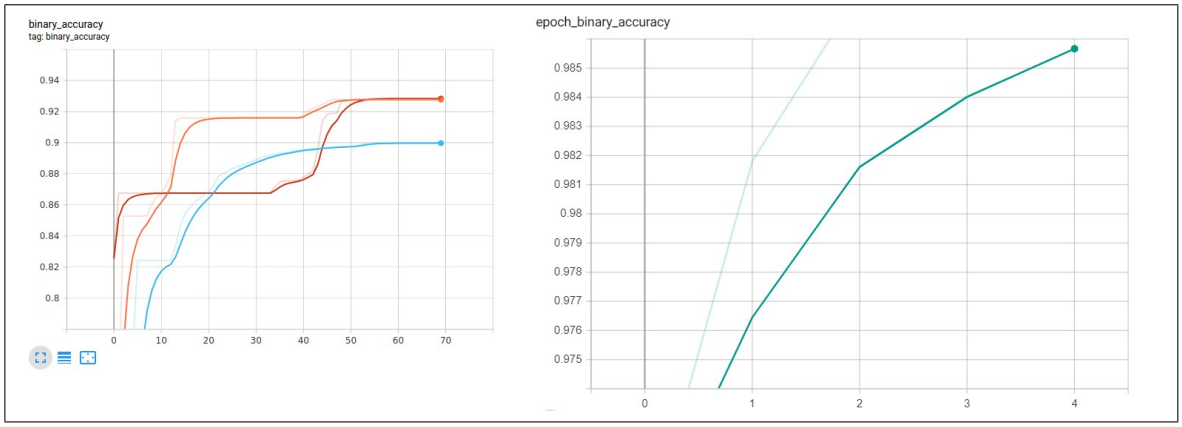

Figure 3.12 Décentralisé vs centralisé

3.4 Expérimentation : IDS 2018

Nous confirmons l'efficacité de l'algorithme fédéré en utilisant un modèle MLP sur une classification multi-classe. Le modèle MLP comporte 83 couches d'entrée et 15 couches de sortie correspondant au nombre d'attaques. Les couches de sortie sont équipées de softmax comme fonction d'activation. Le modèle comporte 4 couches 83, 258, 128 et 64 avec comme fonction d'activation Relu. Étant donné que notre sortie est multiclasse, nous avons choisi de travailler avec une Sparse Categorical Crossentropy comme fonction de perte. Chaque approche aura 100 tours avec le pourcentage de données 70% entraînement et l'ensemble de tests 30%.

Pour évaluer notre modèle, nous avons procédé à la visualisation des métriques sur 100 itérations. Nous avons eu recours aux mêmes stratégies de sélection des clients de l'IDS 2017.

3.4.1 Tous les clients sont sélectionnés

Dans cette sous-section, tous les clients seront sélectionnés dans chaque itération. Puisque nous supposons que toutes les machines seront disponibles et enverront leurs mises à jour à chaque itération, le modèle sera ainsi capable de s'entraîner sur la totalité des données d'entraînement.

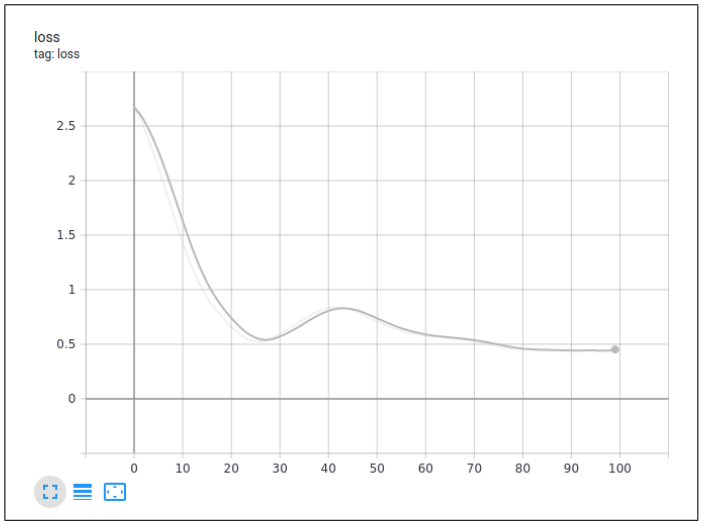

Figure 3.13 Visualisation de la perte de l'apprentissage sur tous les clients IDS2018

Nous pouvons observer une importante diminution de la perte au cours des itérations, ce qui induit un modèle qui s'améliore, la valeur finale est de 0.4.

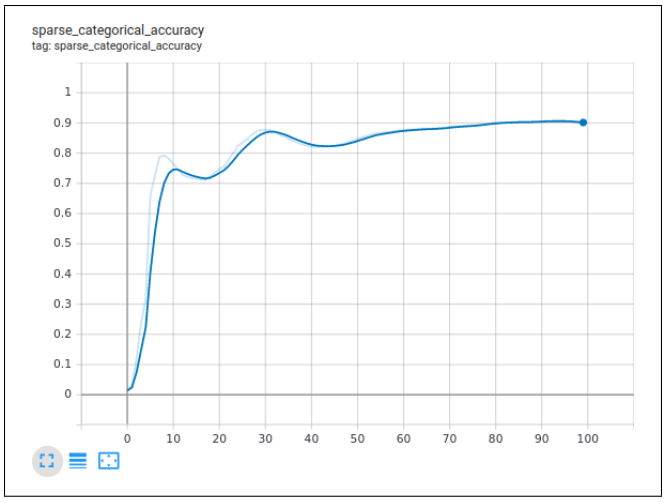

Figure 3.14 Visualisation de la précision de l'apprentissage sur tous les clients IDS2018

Quant à la précision, le modèle prédit de mieux en mieux les valeurs désirées et, puisque tous les clients sont disponibles et qu'il n'y a pas de problème de panne (manque de ressource), il a ainsi pu atteindre une précision de 90%.

3.4.2 Clients aléatoires sélectionnés

Pour cette section, comme pour l'IDS 2017, nous avons opté pour une sélection aléatoire afin de simuler un environnement plus réel dans le cas de perte de connexion, un périphérique qui tombe en panne. Nous constatons, pour la courbe de perte, une grande oscillation due au facteur

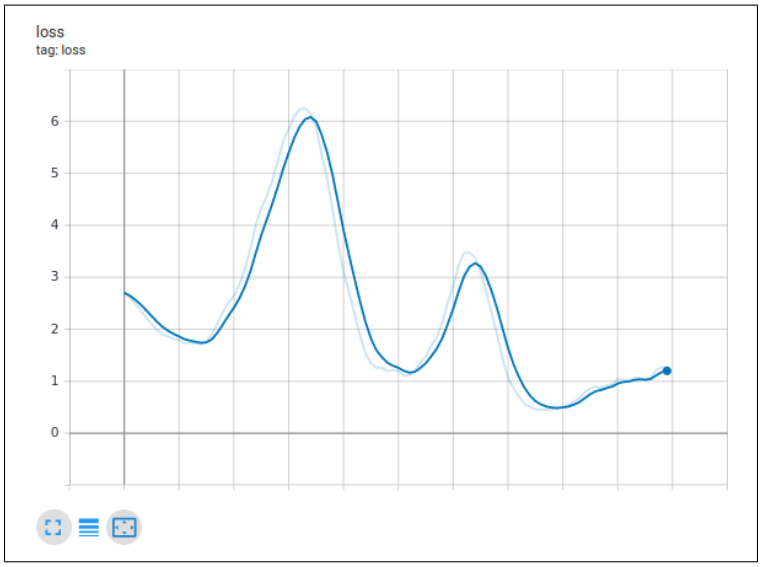

Figure 3.15 Visualisation de la perte sur les clients aléatoires IDS2018

aléatoire du choix des clients. Cependant, nous observons une courbe descendante qui signifie que notre modèle s'améliore au fur et à mesure qu'il s'entraîne.

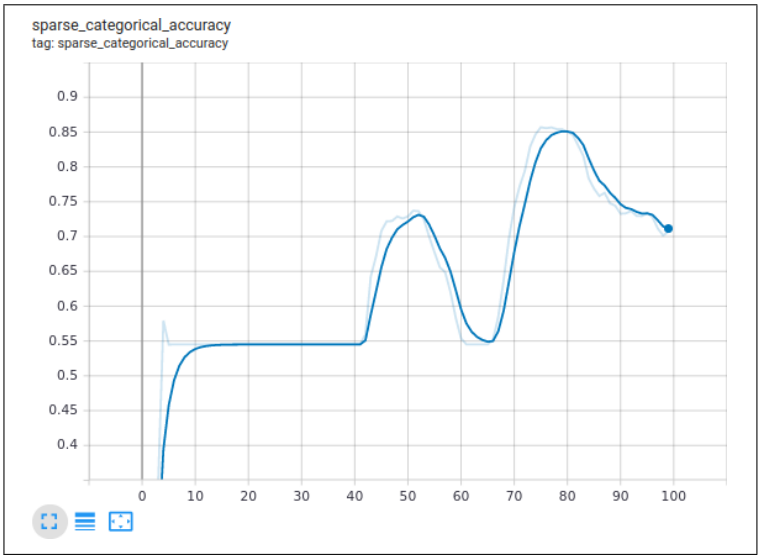

Figure 3.16 Visualisation de la précision de l'apprentissage sur les clients aléatoires IDS2018

Pour la précision, il y a eu initialement une augmentation de cette dernière, mais au final, avec 100 itérations, le modèle rencontre des difficultés à prédire correctement les bonnes étiquettes du fait de l'aspect aléatoire sur lequel repose la sélection des clients.

3.4.3 Sélection des clients en fonction des attaques

Au cours de cette section, nous discuterons les résultats d'expérimentation obtenus lors de la sélection des clients en fonction des attaques. Comme discuté dans la partie IDS2017 de ce chapitre, un seul type d'attaque sera sélectionné à chaque itération. Ainsi, cela nous permet d'expérimenter le comportement de notre modèle dans le cas d'une attaque unique.

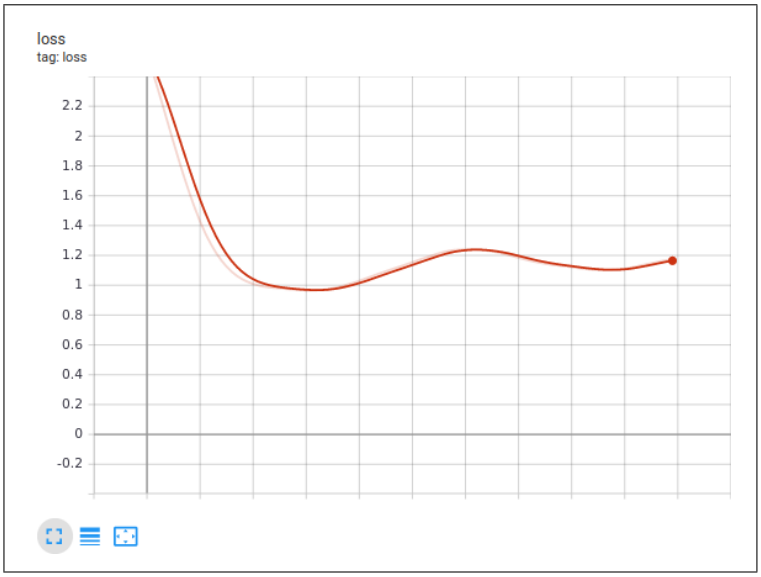

Figure 3.17 Visualisation de la perte sur les clients en fonction des attaques IDS2018

La courbe présente une forme monotone lisse avec une diminution de la perte lors de l'apprentissage, mais avec une valeur élevée de perte égale à 1.1 .

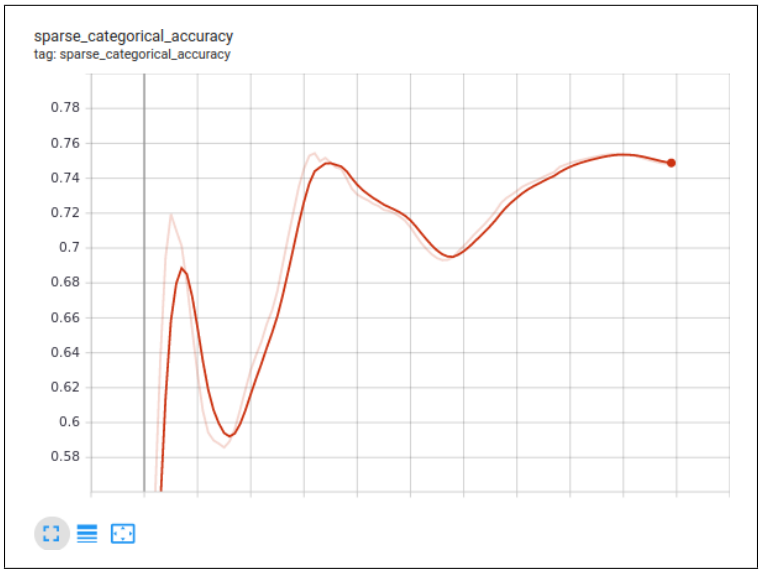

Figure 3.18 Visualisation de la précision de l'apprentissage sur les clients en fonction des attaques IDS2018

Avec cette sélection, le modèle n'a pu atteindre qu'une précision de 75qu'elle a réduit la quantité de données utilisée par le modèle dans la phase d'entraînement.

3.4.4 Sélection en fonction du système d'exploitation

Comme pour l'IDS 2017, cette section se focalisera sur la sélection des clients en fonction du système d'exploitation. Un seul type de système d'exploitation par client sera sélectionné à chaque itération.

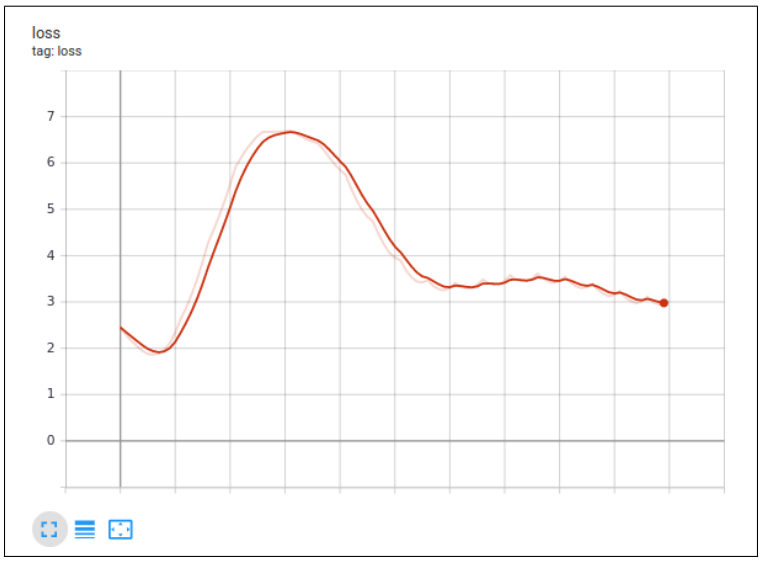

Figure 3.19 Visualisation de la perte sur les clients en fonction des systèmes d'exploitation IDS2018

Dans ce cas, le modèle ne parvient pas à réduire la perte, du fait qu'une seule machine est associée à un système d'exploitation et qu'il est appelé à prédire plusieurs classes d'attaque.

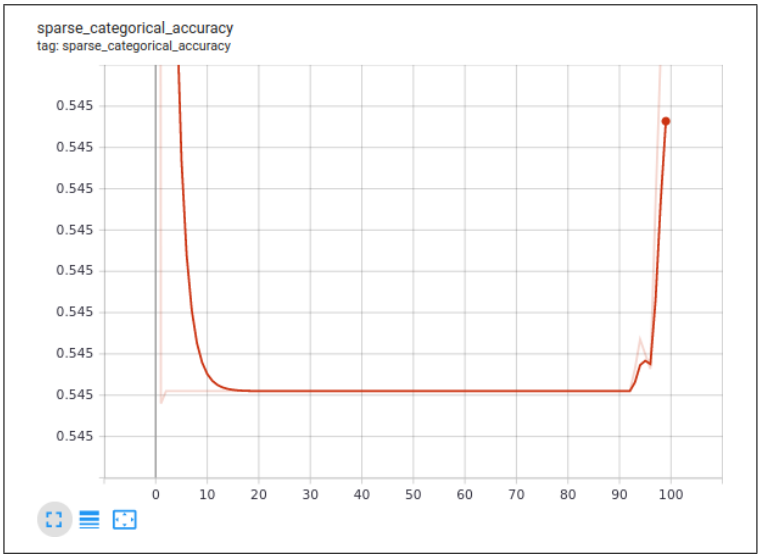

Figure 3.20 Visualisation de la précision de l'apprentissage sur les clients en fonction des systèmes d'exploitation IDS2018

Compte tenu du manque de données et d'une classification plus complexe des instances, le modèle ne parvient pas à produire une prédiction pertinente des résultats et la précision n'a pu atteindre que 0.545.

3.5 Conclusion

Nous avons consacré ce chapitre à l'expérimentation et l'évaluation de notre modèle. Nous avons tout d'abord défini le fractionnement du jeu de données puis présenté les différentes métriques utilisées. Enfin, nous nous sommes focalisés sur l'évaluation du modèle au cours duquel nous avons expliqué les résultats obtenus, les challenges rencontrés et les solutions proposées.

CONCLUSION ET RECOMMANDATIONS

Le sujet s'inscrit dans l'utilisation de l'apprentissage fédéré pour le traitement des intrusions réseau. En effet, cela se base sur la classification des attaques réseau.

Les objectifs de notre projet étaient d'évaluer plusieurs modèles en matière d'apprentissage fédéré, de rechercher et de collecter un jeu de données, de préparer des données, de modéliser et d'évaluer un réseau de neurones pour la détection des attaques.

Notre premier objectif a été atteint dés lors que des modèles ont été étudiés et évalués sur les jeux de données, après avoir récolté les informations nécessaires. Ceci nous a ainsi permis d'identifier les modèles les plus adéquats.

Nous avons ensuite pu atteindre les deux objectifs suivants en nous focalisant sur la collecte de notre jeu de données et sur son pré-traitement.

Le dernier volet de notre projet a été consacré à la modélisation et à l'évaluation des modèles fédérés pour la détection des intrusions réseau. Nous avons ainsi pu aboutir à un modèle CNN fédéré efficace. Cette méthode présente l'avantage de préserver la confidentialité de l'utilisateur puisqu'elle préserve les données côté client.

Dans d'éventuelles nouvelles extensions, nous ambitionnons d'améliorer davantage la précision de nos modèles courants et de parvenir à des modèles plus fitted, susceptibles de détecter plus efficacement les 18 types d'attaques réseau. Le prochain but que nous nous sommes fixé est d'accéder à une sélection intelligente de clients, de reconnaître à l'avance quel client influencera le plus nos modèles pour réduire ainsi le temps d'apprentissage, tout en préservant le même niveau de précision.

ANNEXE I

IMPLÉMENTATION

Au cours de cette annexe, nous nous focalisons sur l'environnement utilisé.

1. Environnement matériel

Dans cette section, nous présentons l'environnement matériel et logiciel.

2. Environnement logiciel

Dans le cadre de la réalisation de notre projet, les ressources auxquelles nous avons eu recours comprennent les équipements suivants :

2.1 Volet de gestion

Tableau-A I-1 Caractéristiques de la machine

Caractéristique	Machine
Système d'exploitation	Ubuntu 18.04.2 LTS
Processeur	Intel® Core™ i7-7700 CPU
Mémoire (RAM)	12Go
Carte graphique	GeForce GTX 1060
Mémoire GPU	8Go
Disk	1TB
SSD	200 Go

2.2 Volet technique

Dans cette partie, nous citons les dépendances techniques à savoir les bibliothèques :

Tableau-A I-2 Logiciels utilisés pour le volet technique

Dépendance	Version	Description
	3.5.6	Python est un langage de programmation interprétée, multiplateforme, fonctionnelle et orientée objet.
	1.5.0	Tensorflow est un framework Google prenant en charge l'exécution de calculs sur le CPU et le GPU, et incluant Tensorflowgpu
	1.5.1	Tensorflow -tensorboard propose une suite d'outils de visualisation proposés par Tensorflow
	1.16.2	NumPy convient à l'informatique scientifique en Python. Il fournit un objet de tableau multidimensionnel et des opérations sur les tableaux.

ANNEXE II

FEDERATED LEARNING FOR ANOMALY-BASED INTRUSION DETECTION

Mohamed Ali Ayed[1], Chamseddine Talhi[1], Hakima Ould-Slimane[1]

[1] Département de génie logiciel et des TI, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

Article publié à la conférence 2021 International Symposium on Networks, Computers and Communications (ISNCC), 25 November 2021.

Federated Learning for Anomaly-Based Intrusion Detection

Mohamed Ali Ayed
École de technologie supérieure
Montreal, Qc, Canada
Email: mohamed-ali.ayed.1@ens.etsmtl.ca

Chamseddine Talhi
École de technologie supérieure
Montreal, Qc, Canada
Email: chamseddine.talhi@etsmtl.ca

Hakima Ould-Slimane
École de technologie supérieure
Montreal, Qc, Canada
Email: cc-hakima.ould-slimane@etsmtl.ca

***Abstract*—We are attending a severe zero-day cyber attacks. Machine learning based anomaly detection is definitely the most efficient defence in depth approach. It consists to analyzing the network traffic in order to distinguish the normal behaviour from the abnormal one. This approach is usually implemented in a central server where all the network traffic is analyzed which can rise privacy issues. In fact, with the increasing adoption of Cloud infrastructures, it is important to reduce as much as possible the outsourcing of such sensitive information to the several network nodes. A better approach is to ask each node to analyze its own data and then to exchange its learning finding (model) with a coordinator. In this paper, we investigate the application of federated learning for network-based intrusion detection. Our experiment was conducted based on the *C*ICIDS2017 dataset. We present a *f* ederated learning on a deep learning algorithm *C*NN based on model averaging. It is a self-learning system for detecting anomalies caused by malicious adversaries without human intervention and can cope with new and unknown attacks without decreasing performance. These experimentation demonstrate that this approach is effective in detecting intrusion.**

I. Introduction

With the rapid growth of information technology, especially forms of communication and social media, there is an ever increasing number of inventions that are being interconnected to more devices which may lead more oftenly to divulged valuable information. Consequently, any individual who uses a PC associated with the Internet is powerless to the dangers that PC programmers and online predators present. Cybersecurity issues are turning into an everyday battle with the resurgence of new attacks. Many devices are having trouble programming automatic updates; they lack reactivity measures like intrusion detection systems (IDS) for detecting vulnerability [1], [2]. Therefore, there is a need for a novel approach for intrusion detection that provides security and apprehends new intrusion attacks as well.

Deep learning methods provide good performance for detecting anomalous events such as having abnormal behaviour. However, in the traditional approach the network data must be collected on each device for server-side machine learning training. This can lead to a security problem or some devices declining the use of their personal data due to user privacy concerns.

This issue is where Federated learning provides a viable solution because only the local models are being aggregated to the server instead of the local data. This type of model can be passively collected by the server in order to update the hyper parameter and have a better detection. Based on the prediction, the network engineers will be able to take necessary action to resolve the issue.

In this article, we are going to present this approach on CICIDS2017 which will be able to identify attacks by aggregating the local models of each device in order to have a better sever-side model.

II. Related work

Detecting anomalies in network traffic has been the spotlight for several years starting with a statistical approach[3], [4], analysing each network packet to apprehend deviations from usual protocol or clustering large numbers of packets [5].

A. Classic machine, learning deeplearning applied for anomaly detection

In the early stages, machine learning [6] and deep learning methods [7], [8], [9] produce intrusion detection systems with good performance in order to detect attacks.

In this article [6], the author evaluates a dataset containing a wealthy mix of normal and abnormal activity against 802.11 networks and multiple machine learning algorthims like AdaBoost and Random Forest. One of the algorithms used to evaluate the dataset, J48 algorithm reached an accuracy rate of 96% which indicate that machine learning is a reliable alternative to detect anomalies in a network.

However, the training data was collected on the server side leading to the disclosure of user privacy and inscreasing the resources used for training. For that reason a new approach is crucial.

B. Federated learning applied for anomaly detection

The concept of federated learning was initially proposed by Google researchers [10]. Their approach was used for image classification tasks and next word prediction tasks.

In our case of anomaly detection in a network, it's more difficult because of the complexity of the network packet and

978-0-7381-1316-6/21/$31.00 ©2021 IEEE

the imbalance issue in the dataset. The CICIDS2017 dataset has a 4:1 ratio of the normal to abnormal instances.

Federated learning in anomaly detection has seen diverse implementation [11], [12], [13], [14], [15] from machine learning to deep neural network in term of security and performance.

As we can see in [16], the authors evaluate their federated model D¨IOT on a dataset collected from various IOT devices in real-world deployment with the help of the Mirai software which carry out large-scale attacks on networks. They conduct their experiment on different numbers of clients ranging from 2 to 15 clients where each client uses a randomized subset of the training dataset with 51 epochs. Even when protecting the privacy of the data, Federated learning detected 94% of attacks.

In [16] the authors show with the increase of devices having more vulnerability, Federated learning is a must and the reliability of of the algorithm to detect the intrusion is thought different dataset. For example Cetin et al. analyzed network attacks from the Aegean Wi-Fi Intrusion Dataset (AWID) reduced dataset and presented the efficiency of the federated AVG algorithm on 10 devices with a precision above 90%.

Federated learning also has two subtypes of learning the federated SGD and the federated AVG. In [10] they demonstrate through a variety of model architectures like LSTM [17], CNN and multi-layer perceptron that the federated avg produces higher quality model in fewer rounds of training.

III. Proposed Approach

In this paper, we propose and evaluate a federated learning method in a network consisting of victims and attackers nodes.

In the next sections, we are going to present the federated learning approach and how to prepare the dataset in order to be used in the experiments.

A. Federated learning Approach

This approach is divided into two parts. The role of the server in every iteration is to share and update the model. This model is what we call the global model. The client side is composed of several machines which may be the victims targeted by Cyberattacks. Each and every machine has its own data consisting of information about the network flow (malicious and benign). Hence, each client will use the labelled data to train a local model without the need to share his sensitive information.

The process is as follows. In each iteration, the clients get a copy of the global model and train it on their local data. Afterwards, one by one they will send the aggregated weights to the server where it is averaged. It is now the turn of server to improve the global model by learning new patterns then distribute the new model to the clients. As time goes by, the model progressively become more efficient.

the purpose of this approach is to predict whether the instance is normal or an attack and stay up to date.

To evaluate the federated learning algorithm, we used a dataset located in the server-side, not seen before by the

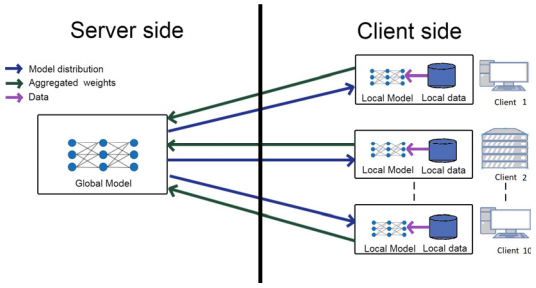

Fig. 1. Federated learning architecture

model, and we test the performance by using the metric Binary Accuracy.

B. Building a Dataset for Federated-Learning-Based Network Intrusion Detection

Experimenting a FL approach and studying its performance is challenging! In fact, the common approach in the FL community is to use a state-of-the-art ML Dataset and simulate its distribution over a set of clients. In such experimentation set-up, researchers use to vary the number of clients and investigate various strategies of distributing samples over them. In the context of Network Intrusion Detection, the experimentation scenarios should be restricted by the network topology, the node characteristics, etc. Indeed, it is not realistic to assign a packet (sample), which is exchanged between a source node A and destination node B, to any machine of the network (C for example)! In addition, the number of possible FL clients is limited by the number of actual nodes belonging to the protected network! Thus the network topology and the actual IP address of each node in the network are required before any distribution of Dataset over a set of learning clients! This will allow a realistic distribution of the original Dataset over the network nodes.

In order to prepare the Dataset for our experimentation, we require the availability of network topology information and source and destination IP addresses in each packet sample of the dataset. Based on these information, we recognize each IP address in the network topology as possible learning client. The distribution of original Dataset packet samples over the learning clients will be as follows:

- If the destination IP address belongs to the network, then a copy of this packet will belong to the 'local' dataset of the client identified by this IP address. In other words, since a network node can 'see' each received packet then it will use it to train its local model.
- If the source IP address belongs to the network, then a copy of this packet will belong to the 'local' dataset of the client identified by this IP address. In other words, since a network node can 'see' each sent packet then it will use it to train its local model.

Based on the aforementioned samples distribution approach,

a packet of the original Dataset can be appear in one or two nodes of the network!

IV. Dataset

The Dataset used to validate the proposed approach was published in 2017 by the Canadian Institute for Cyber Security and untitled 'IDS2017' [18].The network topology is composed of two networks (Fig. 2). The first network is the victim network which is composed of 6 desktops, 1 laptop and 3 servers with different installed operating systems and related public and private IP addresses. The second network is the attack-network which is composed of 4 desktops with Kali or Windows 8.1 as operating systems. Each network is connected to the Internet through a switch-router equipment. The dataset consists of a network traffic collected throughout 5 days : from July 3 to July 7, 2017. The classes of traffic collected each day are listed in Table.1. The traffic can be either normal or malicious.

Days	Labels
Monday	Benign
Tuesday	BForce/SFTP/SSH
Wednesday	Web and Infiltration Attacks Web BForce, XSS and Sql Inject. Infiltration Dropbox Download and Cool disk
Thursday	Web and Infiltration Attacks Web BForce, XSS and Sql Inject. Infiltration Dropbox Download and Cool disk
Friday	DDoS LOIT, Botnet ARES, PortScans (sS,sT,sF,sX,sN,sP,sV,sU, sO,sA,sW,sR,sL and B)

Table. 1. Data labels.

The cyberattacks belonging to the IDS2017 Datastet malicious traffic are described below :

- Bruteforce attacks: the attacker is a Kali Linux machine having the IP address 205.174.165.73 and the victim is the web server running an Ubuntu 16.04 operating system and having the IP address 192.168.10.50.
- DoS attack: the attacker is a Kali Linux having the IP address 205.174.165.73 and the victim is an Ubuntu 12.04 machine having the IP address 192.168.10.51.
- Web attack: The attacker is a Kali Linux having the IP address 205.174.165.73 and the victim is web server running an Ubuntu 16.04 operating system and having the IP address 192.168.10.50.
- Infiltration attack: The attacker is a Kali Linux machine having the IP address 205.174.165.73 and the victims are various machines running various operating systems: Windows, Ubuntu and Macintosh.
- Botnet attack: The attacker is a Kali Linux having the IP address 205.174.165.73 and the victims are Windows machines: running Windows Vista(192.168.10.8), Windows 7(192.168.10.9), Windows 8.1(192.168.10.5), or Windows 10 (192.168.10.14).
- DDoS attack and PortScan: The attackers are the 3 desktops in the attack-network running the Windows 8.1 operating systems and having the IP addresses 205.174.165.69, 205.174.165.70, and 205.174.165.71. The victim machine is the web server running the Ubuntu 16 operating system and having the IP address 192.168.10.50 as well as all the windows machines: IP addresses 192.168.10.8, 192.168.10.9, 192.168.10.5, and 192.168.10.14.

The number of normal and attack samples (packets) is shown in Table.2.

Instance	Total
BENIGN	2104911
Bot	1966
DDoS	128027
DoS GoldenEye	10293
DoS Hulk	231073
DoS Slowhttptest	5499
DoS slowloris	5796
FTP-Patator	7938
Heartbleed	11
Infiltration	36
PortScan	158930
SSH-Patator	5897

Table. 2. Number of instance.

The dataset contains records labeled "benign" and multiple types of attacks like Brute Force FTP, Brute Force SSH, DOS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. We grouped these attacks into one class labelled 'attack' to perform binary classification (normal vs abnormal). By applying the approach detailed in subsection III. B, we have identified each machine of the network as client and assigned to each client a subset of the IDS2017 Dataset samples. The number of samples assigned to each client is presented in Table. 3.

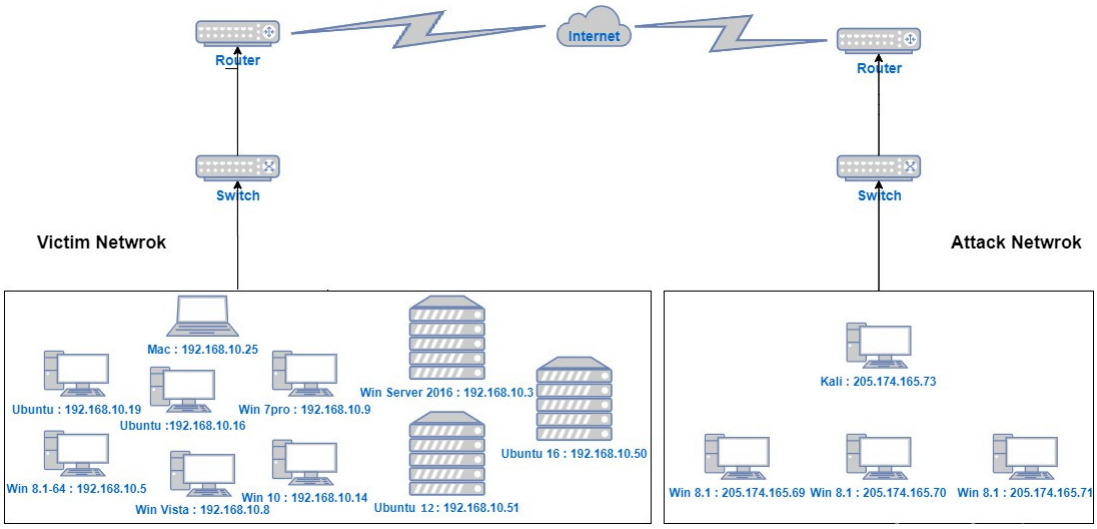

Fig. 2. IDS2017 Network [18]

Client	IP address	Label	Total rows
0	192.168.10.3	benign	905365
0	192.168.10.3	attack	0
1	192.168.10.50	benign	105395
1	192.168.10.50	attack	552373
2	192.168.10.51	benign	96360
2	192.168.10.51	attack	11
3	192.168.10.19	benign	188693
3	192.168.10.19	attack	0
4	192.168.10.16	benign	320553
4	192.168.10.16	attack	0
5	192.168.10.9	benign	177716
5	192.168.10.9	attack	372
6	192.168.10.5	benign	211680
6	192.168.10.5	attack	287
7	192.168.10.8	benign	236163
7	192.168.10.8	attack	407
8	192.168.10.14	benign	327407
8	192.168.10.14	attack	922
9	192.168.10.25	benign	189928
9	192.168.10.25	attack	0

Table. 3. Distribution of IDS2017 Samples over FL Clients.

We balanced the prepossessed dataset. The resulting dataset has the same number of normal and attack samples (554374) [19]. This way, the trained model will be able to learn the pattern of each class equally.

V. PERFORMANCE EVALUATION

In our experiments, we apply federated learning approach to the IDS2017 dataset.

In the next subsections, we are going to discuss the efficiency of the CNN model and federated learning as well as a comparison between the centralized and the decentralized approach.

A. Efficiency of CNN model

To produce an efficient CNN model, we went throught several steps:

First, we combine all the ML dataset in order to have attacks thoughout the days.

Second, we change the label from categorical data (Benign, attacks) to numerical data (0,1) because most of the Machine learning algorithms cannot handle categorical variables unless we convert them to numerical values.

Third, we divide the dataset into test and train subsets so that we can later evaluate the efficiency of the model on the test subset. 70% will be used for training the model and 30% for the testing.

Fourth, we need to scale the train, test dataset. We chose to use the standardscaler to transform the dataset such that its distribution will always have a mean value 0 and standard deviation of 1. That way, the standardscaler will make our variable contribute equally to the model fitting and the model learning functions.

For the centralized CNN model, we chose a batch_size equal to 1000, 5 epochs and a filter size of 3 .

Considering that we used the combined dataset, the model has the following characteristics :

- 78 input layers and 1 output layer.
- 3 convolution layers, each using a batch normalization, a MaxPooling1D with strides equal to 1 and a droprate of 0.5.

- 3 fully connectedlayers equiped with batch normalizationn droprate and Relu as an activation function.
- the output layer having a sigmoid as an activation function.

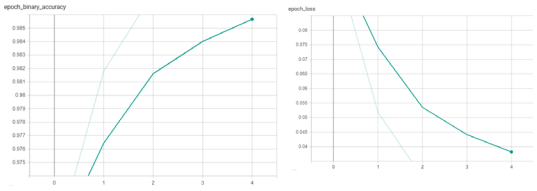

Fig. 3. CNN model prediction rate.

As shown in figure 3, we can see that the CNN model is highly efficient in detecting network intrusion by having an accuracy of 0.988%.

B. *Efficiency of Federated Learning*

We confirm the effectiveness of the federated algorithm by using a CNN model.

The model has 74 input layers and 1 output layer corresponding respectively to the number of attributes and the number of classes. The output layers are equipped with softmax as an activation function. The model has 3 convolutional layers and 3 Fully Connected Layers. The neurons are equipped with the Relu activation function . Considering that we have a binary output we chose to work with a Binary cross entropy as a loss function.

We utilize a filter of size 3 and a learning rate of 0.5, batch sizes of 32 and 64, 7 epochs and 8 devices per round. Every epoch will have 10 rounds with each one using only 10% of the subset of each device. With every experiment, we changed the percentage of data used as a training set for each device to 60,70 and 80% and the test set to 40,30 and 20 as well as the client selection in each round.

1) All clients being selected: The convergences behaviours of the Federated algorithm on the parameter listed above are shown in Figure 4, 5 , 6 and 7 with all the client being available in each round. We also exhibit the comparison between those parameters and the centralized approach in Figure 8.

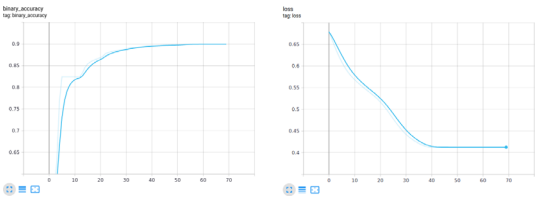

Fig. 4. 60 % train batch 32.

Trials with a 60% train and a batch size of 32 led us to an accuracy rate of 90% and a loss rate of 42%.

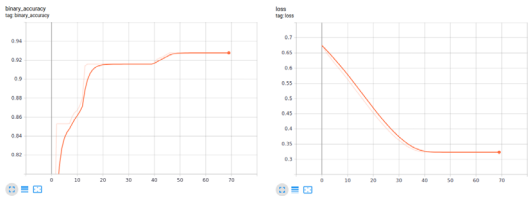

Fig. 5. 70 % train batch 32.

Experimenting With a 70% train and a batch size of 32, resulted in an accuracy rate of 93% and a loss rate of 32%.

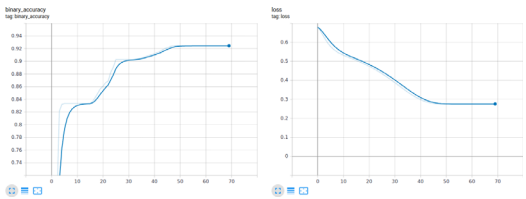

Fig. 6. 70 % train batch 64.

With a 70% train and a batch size of 64, we have an accuracy rate of 92.5% and a loss rate of 28%.

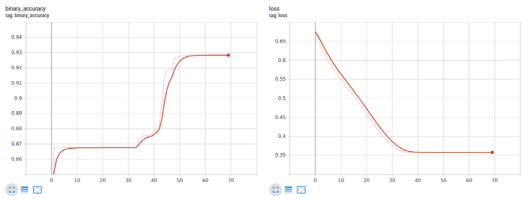

Fig. 7. 80 % train batch 32.

Testing on an 80% train and a batch size of 32, we have an accuracy rate of 92.9% and a loss rate of 36%.

Consequently we found out that with a 70% train and a batch size of 32 we have the best accuracy with a rate of 93%.

2) random clients being selected: With a more real scenario where certain devices can go down, we chose to select a random number of clients each round. The result of the Federated algorithm on the parameters listed above are shown in Figure 8, 9 and 10.

Experimenting with a random client being selected in each round with a 60% train and a batch size of 32. Led us to an accuracy rate of 72% and a loss rate of 48%.

Testing With random client being available in each round with a 70% train and a batch size of 32, have resulted in having an accuracy rate of 91% and a loss rate of 23%.

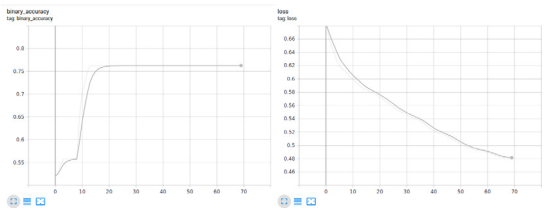

Fig. 8. 60 % train batch 32 with random client.

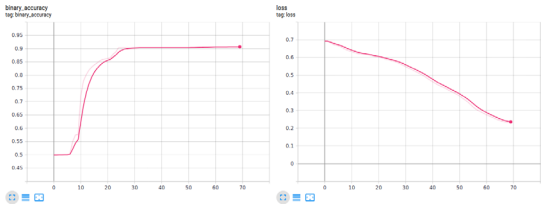

Fig. 9. 70 % train batch 32 with random client.

The experiment on a 80% train and a batch size of 32 with random client, we obtained an accuracy rate of 90% and a loss rate of 33%.

Consequently we discovered that even with a random selection of clients in each round, the parameter of a 70% train and a batch size of 32 has the best accuracy with a rate of 91%.

3) client selection depending on the attacks: In this section, The client selection is different from one round to another depending on the type of the attack:

- Round 1 : focusing on the client being attacked by brute-force which is client 1 with IP address 192.168.10.50
- Round 2 : training on the clients being attacked by Dos attack which are client 1 and 4 with IP address 192.168.10.50 and 192.168.10.16
- Round 3 : the client selected is the target of web attacks which is client 1 with IP address 192.168.10.50
- Round 4 : the federated algorithm selects the clients with an infiltration attack which are all clients except client 0, 1 and 2
- Round 5 : Training on the client with Botnet attack which are clients 5, 6, 7 and 8 with IP address 192.168.10.9, 192.168.10.5, 192.168.10.8, 192.168.10.14 .
- Round 6 : Focusing on the client being attacked by DDos attack and Portscan which is client 1 with IP address 192.168.10.50.
- Round 7 : The federating learning algorithm will be trained by selecting all the clients.

Examining with the client selection discussed above with a 60% train and a batch size of 32 provides us with an accuracy rate of 91% and a loss rate of 52%.

The trial with a 70% train and a batch size of 32, providse us with an accuracy rate of 93% and a loss rate of 31%.

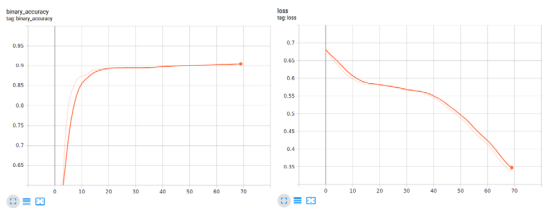

Fig. 10. 80 % train batch 32 with random client.

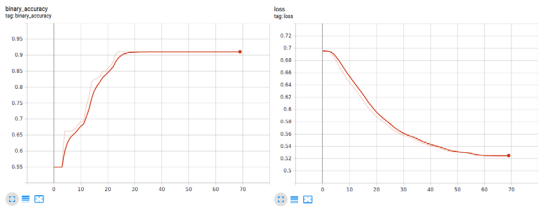

Fig. 11. 60 % train batch 32 with type of attacks client.

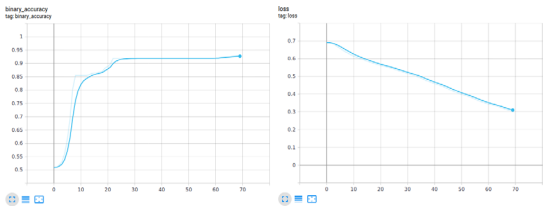

Fig. 12. 70 % train batch 32 with type of attacks client.

Experimenting on an 80% train and a batch size of 32 with the chosen clients depending on the attacks, we have an accuracy rate of 90% and a loss rate of 43%.

Therefore, we found that even with the selection of the clients in each round the parameter with a 70% train and a batch size of 32 have the best accuracy rate of 93%.

4) client selection depending on the operating system: In this section, the client selection is focused on the operating systems available in the network where in each round we take one type of operating system:

- Round 1 : The clients selected will be 6, 3, 1, 9 and 0 with IP address 192.168.10.5, 192.168.10.19, 192.168.10.50, 192.168.10.25 and 192.168.10.3
- Round 2 : The operating system selected in this round will be windows 7 pro client 5, Ubuntu client 4, Ubuntu 12 client 2, Mac client 9 and Win server 2016 client 0 with IP address 192.168.10.19, 192.168.10.16
- Round 3 : Training on the clients 8,3,1,9 and 0 with IP address 192.168.10.14, 192.168.10.19, 192.168.10.50, 192.168.10.25 and 192.168.10.3
- Round 4 : The federated algorithm selects the client 7,4,2,9 and 0 with IP address 192.168.10.8,

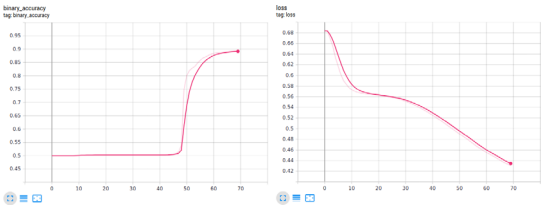

Fig. 13. 80 % train batch 32 with type of attacks client.

192.168.10.16, 192.168.10.51, 192.168.10.25 and 192.168.10.3

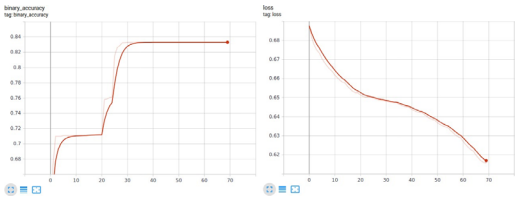

Fig. 14. 60 % train batch 32 with operating system client.

Testing on the clients with differents operating systems and having 60% train and a batch size of 32 supplies us with an accuracy rate of 81% and a loss rate of 53%.

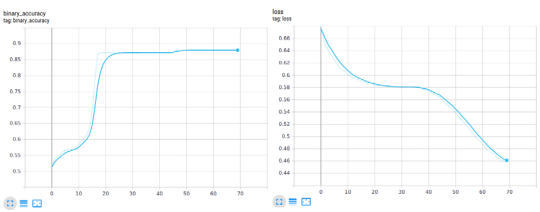

Fig. 15. 70 % train batch 32 with operating system client.

Training the federated algorithm on 70% train and a batch size of 32, has come about to having an accuracy rate of 89.5% and a loss rate of 39%.

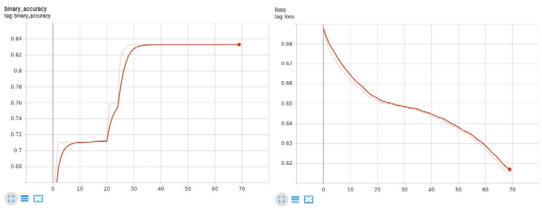

Fig. 16. 80 % train batch 32 with operating system client.

Experimenting on a 80% train and a batch size of 32 with the chosen clients, we have an accuracy rate of 83% and a loss rate of 61%.

Subsequently, we determine that with the choice of the clients with different operating systems in each round, the parameter 70% train and a batch size of 32 has the best accuracy with a rate of 88%.

5) Comparison between the four approaches: After experimenting with the four approaches of client selection on different sections of the dataset, we figured out that the federated learning algorithm is reliable even if not all the clients are available in each round due to power going down or loss of connection.

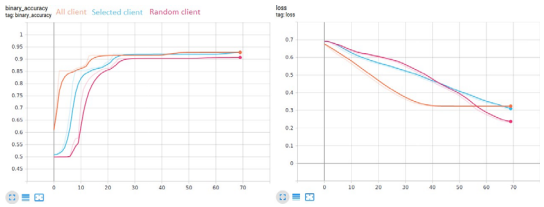

Fig. 17. Comparison accuracy rate and loss.

As we can see in figure 17, the accuracy of the federated algorithm with client selection depending on the type of the attack is the same as all the clients being selected in each round.

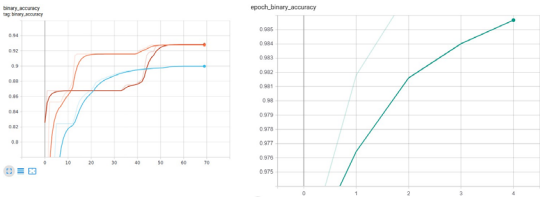

Fig. 18. decentralized vs centralized.

In the figure 18, the result of the Federated learning accuracy is lower then the accuracy of the training in a centralized method. However, it provides more privacy for clients because it doesn't require their sensitive information as training data.

VI. Conclusion

In this paper, we have proposed a new approach to detect intrusion in the network with federated learning using CICIDS2017 dataset.

In fact, the classical deep learning approach proposes a model that requires data to be transferred to the server side compared to the proposed approach which use the federated learning model built upon deep learning which leaves the data in the client side. Therefore this method has the advantage of preserving user privacy [20], [21], [22] while performing similarly to the classical deep learning.

As part of future research work:

- The model will have multiple labels as outputs for the attacks instead of just using a binary classification therefore it will be able to differentiate the attacks from one another.
- Run extra tests to show the adequacy of federated learning for the network intrusion detection problem.
- Try running the model on real world dataset.

References

[1] J. Sen and S. Mehtab, "Machine learning applications in misuse and anomaly detection," in *Security and Privacy From a Legal, Ethical, and Technical Perspective* (C. Kalloniatis and C. Travieso-Gonzalez, eds.), ch. 10, Rijeka: IntechOpen, 2020.

[2] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Hyperspherical cluster based distributed anomaly detection in wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 74, p. 1833–1847, 01 2014.

[3] M. Ahmed, A. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 11 2015.

[4] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 12 2015.

[5] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," 11 2001.

[6] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Communications Surveys and Tutorials*, vol. 18, pp. 184–208, 2016.

[7] S. Wang, B. Li, M. Yang, and Z. Yan, *Intrusion Detection for WiFi Network: A Deep Learning Approach: 11th EAI International Conference, WiCON 2018, Taipei, Taiwan, October 15-16, 2018, Proceedings*, pp. 95–104. 01 2019.

[8] J. Ran, Y. Ji, and B. Tang, "A semi-supervised learning approach to ieee 802.11 network anomaly detection," pp. 1–5, 04 2019.

[9] V. Thing, "Ieee 802.11 network anomaly detection and attack classification: A deep learning approach," pp. 1–6, 03 2017.

[10] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016.

[11] Q. Qin, K. Poularakis, K. K. Leung, and L. Tassiulas, "Line-speed and scalable intrusion detection at the network edge via federated learning," in *2020 IFIP Networking Conference (Networking)*, pp. 352–360, 2020.

[12] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "Deepfed: Federated deep learning for intrusion detection in industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[13] X. Hei, X. Yin, Y. Wang, J. Ren, and L. Zhu, "A trusted feature aggregator federated learning for distributed malicious attack detection," *Computers and Security*, vol. 99, p. 102033, 2020.

[14] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," pp. 374–380, 08 2019.

[15] Y. Sun, H. Ochiai, and H. Esaki, "Intrusion detection with segmented federated learning for large-scale multiple lans," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2020.

[16] T. D. Nguyen, S. Marchal, M. Miettinen, M. H. Dang, N. Asokan, and A. Sadeghi, "D¨iot: A crowdsourced self-learning approach for detecting compromised iot devices," *CoRR*, vol. abs/1804.07474, 2018.

[17] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," 04 2015.

[18] I. Sharafaldin, A. Habibi Lashkari, and A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," pp. 108–116, 01 2018.

[19] J. Ran, Y. Ji, and B. Tang, "A semi-supervised learning approach to ieee 802.11 network anomaly detection," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pp. 1–5, 2019.

[20] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Oct 2016.

[21] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Privacy aware learning," 2013.

[22] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, p. 211–407, Aug. 2014.

[23] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016.

[24] B. Cetin, A. Lazar, J. Kim, A. Sim, and K. Wu, "Federated wireless network intrusion detection," in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 6004–6006, 2019.

[25] F. Sattler, S. Wiedemann, K. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *CoRR*, vol. abs/1903.02891, 2019.

[26] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Convergence of update aware device scheduling for federated learning at the wireless edge," 2020.

[27] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2020.

RÉFÉRENCES

Alazab, M., Priya, S., M, P., Reddy, P., Gadekallu, T. & Pham, Q.-V. (2021). Federated Learning for Cybersecurity : Concepts, Challenges and Future Directions. *IEEE Transactions on Industrial Informatics*. doi : 10.1109/TII.2021.3119038.

Azencott, C.-A. (2020, Février). Jeu de données. Repéré à https://openclassrooms.com/fr/courses/4297211-evaluez-les-performances-dun-modele-de-machine-learning/4308241-mettez-en-place-un-cadre-de-validation-croisee.

Beaugnon, A. (2018). *Expert-in-the-Loop Supervised Learning for Computer Security Detection Systems*. (Theses, PSL Research University). Repéré à https://hal.archives-ouvertes.fr/tel-01888971.

Campos, E. M., Saura, P. F., González-Vidal, A., Hernández-Ramos, J. L., Bernabe, J. B., Baldini, G. & Skarmeta, A. (2021). Evaluating Federated Learning for Intrusion Detection in Internet of Things : Review and Challenges.

Chatterjee, S. & Hanawal, M. K. (2021). Federated Learning for Intrusion Detection in IoT Security : A Hybrid Ensemble Approach.

Datamok. [https ://fr.wikipedia.org/wiki/Sensibilit%C3%A9*ets p%C*3%*A*9*ci f icit%C*3%*A*9]. (2021*, Dcem*

de l Europe, C. [https ://www.coe.int/fr/web/artificial-intelligence/glossary]. (2020, Février). Apprentissage automatique.

Haghighat, M. H., Foroushani, Z. A. & Li, J. (2019, Oct). SAWANT : Smart Window Based Anomaly Detection Using Netflow Traffic. *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, pp. 1396-1402. doi : 10.1109/ICCT46805.2019.8947103.

Haghighat, M. H., Foroushani, Z. & Li, J. (2019, 10). SAWANT : Smart Window Based Anomaly Detection Using Netflow Traffic. pp. 1396-1402. doi : 10.1109/ICCT46805.2019.8947103.

Herschel, M., Diestelkämper, R. & Ben Lahmar, H. (2017). A Survey on Provenance : What for ? What Form ? What From ? *The VLDB Journal*, 26(6), 881–906. doi : 10.1007/s00778-017-0486-1.

Kang, N. (2017, Juin). Un neurone. Repéré à https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f.

Kim, J., Kim, J., Thu, H. L. T. & Kim, H. (2016, Feb). Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection. *2016 International Conference on Platform Technology and Service (PlatCon)*, pp. 1-5. doi : 10.1109/PlatCon.2016.7456805.

Maurice, B. (2018, Septembre). Évaluation du modèle. Repéré à https://deeplylearning.fr/tag/ sur-apprentissage/.

Miao, H., Li, A., Davis, L. S. & Deshpande, A. (2017, April). Towards Unified Data and Lifecycle Management for Deep Learning. *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pp. 571-582. doi : 10.1109/ICDE.2017.112.

of new brunswick, U. [https ://www.unb.ca/cic/datasets/ids-2018.html]. (2018, Février). IDS2018.

Pillou, J.-F. (2018, Mars). Attaque sur les réseaux informatique. Repéré à https://www.commentcamarche .net/contents/47-piratage-et-attaques-informatiques.

Pimentel, J. a. F., Freire, J., Murta, L. & Braganholo, V. (2019). A Survey on Collecting, Managing, and Analyzing Provenance from Scripts. *ACM Comput. Surv.*, 52(3). doi : 10.1145/3311955.

Polyzotis, N., Roy, S., Whang, S. E. & Zinkevich, M. (2018). Data Lifecycle Challenges in Production Machine Learning : A Survey. *SIGMOD Rec.*, 47(2), 17–28. doi : 10.1145/3299887.3299891.

Raycad. (2017, Novembre). Un réseau de neurones. Repéré à https://medium.com/@raycad.seedotech/ convolutional-neural-network-cnn-8d1908c010ab.

Sarhan, M., Layeghy, S., Moustafa, N. & Portmann, M. (2021). A Cyber Threat Intelligence Sharing Scheme based on Federated Learning for Network Intrusion Detection.

Sharafaldin, I., Habibi Lashkari, A. & Ghorbani, A. (2018, 01). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. pp. 108-116. doi : 10.5220/0006639801080116.

Souza, R., Azevedo, L., Lourenço, V., Soares, E., Thiago, R., Brandão, R., Civitarese, D., Brazil, E., Moreno, M., Valduriez, P., Mattoso, M., Cerqueira, R. & Netto, M. (2019, Nov). Provenance Data in the Machine Learning Lifecycle in Computational Science and Engineering. *2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*, pp. 1-10. doi : 10.1109/WORKS49585.2019.00006.

Subramaniam, V. (2019a, Juillet). Choix du modèle. Repéré à https://medium.com/secure-and-private -ai-writing-challenge/federated-learning-an-introduction-93bc0167f916.

Subramaniam, V. (2019b, Juillet). Diffusion du modèle. Repéré à https://medium.com/secure-and -private-ai-writing-challenge/federated-learning-an-introduction-93bc0167f916.

Subramaniam, V. (2019c). Entraînement du modèle par les périphériques. Repéré à https://medium.com/ secure-and-private-ai-writing-challenge/federated-learning-an-introduction-93bc0167f916.

Subramaniam, V. (2019d, Juillet). Agrégation des gradients. Repéré à https://medium.com/secure-and-private-ai-writing-challenge/federated-learning-an-introduction-93bc0167f916.

Wang, F., Zhu, H., Tian, B., Xin, Y., Niu, X. & Yang, Y. (2011). A HMM-based method for anomaly detection. *2011 4th IEEE International Conference on Broadband Network and Multimedia Technology*, pp. 276-280. doi : 10.1109/ICBNMT.2011.6155940.

Wang, Y., Wong, J. & Miner, A. (2004). Anomaly intrusion detection using one class SVM. *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pp. 358-364. doi : 10.1109/IAW.2004.1437839.

Xu, G., Li, H., Liu, S., Yang, K. & Lin, X. (2020). VerifyNet : Secure and Verifiable Federated Learning. *IEEE Transactions on Information Forensics and Security*, 15, 911-926. doi : 10.1109/TIFS.2019.2929409.

Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F. & Zumar, C. (2018). Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.*, 41, 39-45.

Zhang, M., Xu, B. & Gong, J. (2015). An Anomaly Detection Model Based on One-Class SVM to Detect Network Intrusions. *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pp. 102-107. doi : 10.1109/MSN.2015.40.

Zhang, Z., Sparks, E. R. & Franklin, M. J. (2017). Diagnosing Machine Learning Pipelines with Fine-Grained Lineage. *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*, (HPDC '17), 143–153. doi : 10.1145/3078597.3078603.

Zhu, G. (2020, Février). Apprentissage fédéré. Repéré à TowardsanIntelligentEdge: WirelessCommunicationMeetsMachineLearning-ScientificFigureonResearchGate .Availablefrom:https://www.researchgate.net/figure/a-Federated-learning-using-wirelessly-distributed-data-b-Performance-comparison_fig2_327434818[accessed25Feb,2020].