Detection of Obfuscation Malware: A Federated Transfer Learning-based Approach with Hybrid Neural Networks

Carlos J. Reis, Carlos A. C. Tojeiro, Thiago José Lucas, and Kelton A. P. da Costa, *Senior Member, IEEE*

***Abstract*—The increase in the incidence of cyberattacks, especially through the use of complex mechanisms for exploiting vulnerabilities, such as malware obfuscation, has driven the adoption of Machine Learning (ML) techniques in cybersecurity. This study investigates the application of Federated Learning (FL), a decentralized approach that preserves data privacy and overcomes challenges in transferring large volumes of information. Furthermore, it proposes an innovative solution for the application of Transfer Learning (TL) techniques, integrating a DevOps pipeline. Two labeled datasets were used, CIC-MalMem-2022 and Malware Detection Dataset, along with two FL frameworks, Flower Framework and TensorFlow Federated. A decentralized model based on a Linear Neural Network (LNN) with federated averaging (FedAvg) was compared to a centralized model using a Recurrent Neural Network (RNN) in supervised binary classifications of malware. The results demonstrate high accuracy across all analyzed scenarios, highlighting the outcomes obtained in centralized training for the CIC-Malware dataset, achieving an accuracy of 0.99, precision of 1.0, and recall of 0.99, emphasizing the potential of FL in cybersecurity.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/9689

***Index Terms*—Federated learning, Transfer learning, Federated transfer learning, malware, cyberattacks, neural network, TensorFlow.**

I. Introduction

As attacks on computing systems and computer networks become increasingly complex, the need to develop sophisticated intrusion detection methods is becoming more evident every day. Malware poses a significant threat to the confidentiality, integrity, and availability of personal, corporate, or government data. Generally, three types of malware have been frequently used by crackers: Spyware, Ransomware, and Trojan Horse [1]. Spyware employs techniques to access sensitive or confidential information to collect data, being regarded as the type of malware that most threatens user privacy [2]. On the other hand, Ransomware has a unique characteristic that, regarding privacy, allows command and control communication with the attacker's workstation, facilitating data exfiltration (illegally extracting and transferring confidential information) prior to encryption [3]. Finally, Trojan Horses can install backdoors [1] that enable surveillance and data theft while misleading the user into making mistakes by pretending to be a legitimate application [4].

Although each category of privacy-targeted malware operates with distinct mechanisms and architectures, many of today's families of spyware, ransomware, and Trojans share a common feature: once they are executed by a legitimate operating system process, they tend to be undetectable by security controls until the malware has fully or partially achieved its objective [5]. This occurs mainly because they use obfuscation techniques as an evasion method. Generally, current malware detection techniques focus on recognizing unobfuscated malware, and so far, there is no clear and unified technique for detecting and understanding the patterns and behaviors of obfuscated malware in real-time [1], [6].

Global models in FL face significant challenges during the training process, particularly due to obfuscation techniques employed by malware, such as packaging, encoding, and encryption. To mitigate these obstacles, TL based approaches have been adopted, with fine-tuning strategies carried out on the client side. These solutions aim to enhance the generalization capability of the global model, even in contexts with heterogeneous data [7].

For this reason, machine learning algorithms, especially LNNs, are frequently utilized to analyze obfuscated privacy malware.

The main scientific contributions of this work are an innovative approach that combines Federated Learning (FL) and Transfer Learning (TL) for malware detection, ensuring data privacy. By utilizing real datasets and frameworks such as Flower and TensorFlow Federated, the models achieved robust performance (both centralized and decentralized), demonstrating the feasibility of FL in cybersecurity in a scalable and secure manner.

Recently, there has been a significant increase in the sophistication of cyber attacks, particularly with the use of

The associate editor coordinating the review of this manuscript and approving it for publication was Giner Alor-Hernández (*Corresponding author: Carlos de Jesus Reis*).

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES).

Carlos de Jesus Reis, C. A. C. Tojeiro, and K. A. P. da Costa are with the Department of Computing, São Paulo State University, Bauru - SP, Brazil (e-mails: cj.reis@unesp.br, carlos.tojeiro@unesp.br, and kelton.costa@unesp.br).

T. J. Lucas, is with the Department of Information Security São Paulo State College of Technology, Av. Vitalina Marcusso 1400, Ourinhos - SP, Brazil (e-mail: thiago@fatecourinhos.edu.br).

[1] A backdoor is a method by which authorized and unauthorized users can bypass security measures to gain the highest level of permission (Administrator) on a computer system, network, or software application.

obfuscated malware. According to data from CERT.br, in 2024, over 1.2 million security incident notifications were recorded, representing a 15% increase compared to the previous year. Among these notifications, there is notable growth in attacks related to obfuscated malware, which complicates [8] identification and mitigation efforts by security teams.

A study by Symantec (Broadcom) indicated that 93% of malicious files detected in 2022 were distributed via email using obfuscation techniques, primarily in JavaScript scripts or Office documents with malicious macros [9].

This work presents a detection system for obfuscated malware that achieves competitive metrics using advanced ML techniques, focusing on a binary scenario that allows the distinction between processes displaying patterns and behaviors associated with malicious infections, utilizing a logistic regression model.

The overall objective of this research is to accurately detect malicious network attacks, employing FTL to preserve the privacy and security of institutional data. The LNN model will be shared among cores, organizations, and edge devices, maintaining its effectiveness.

The configuration includes edge devices, IoT, three organizations (users), and a cloud server that hosts the global model, with the possibility of expansion to more organizations as needed. Additionally, a comparison was made between two TL Frameworks during the experiments, and a DevOps framework was implemented to assist in transferring the pre-trained model to new architectures. In this context, this paper presents an innovative approach that combines FL, TL techniques, and hybrid models for the detection and mitigation of obfuscated malware, contributing to the cybersecurity of corporate networks.

The remainder of this article is organized as follows: Section II presents a review of related works in the area of detection and mitigation of obfuscated malware in edge devices using FL. Section III provides a brief overview of some basic concepts. Section IV proposes the FL method for the detection and mitigation of malware, along with the performance metrics used. Section V discusses the results, and the conclusions are presented in Section VI.

II. Related Work

The problem addressed in this paper has been widely investigated in the literature, given its relevance. Various approaches have been proposed, seeking solutions for malware classification using Federated Learning Frameworks.

In the analysis of the article by Karimireddy et al. [10], the authors relate the main characteristics to identify a good Federated Learning framework, highlight the need for a systematic analysis of FL frameworks, motivated by researchers' decision-making, and identify suitable frameworks and future research gaps. Thus, they analyze 14 FL frameworks (TensorflowFederated, PySyft, FATE, PaddleFL, FedBioMed, Substra, FedML, FLower, FederatedScope, OpenFL, NVFLARE, APPFL, Vantage6, and FedN), focusing on characteristics such as data distribution, communication topologies, and security. Therefore, the authors conclude that FedML and FederatedScope stand out for their comprehensive functionalities.

In the study by Cevallos-Salas et al. [11], the authors conduct research on the detection and classification of obfuscated privacy malware, using memory dump analysis techniques. The authors develop three classifiers: a binary classifier that distinguishes between benign and malicious samples using logistic regression, and two multiclass classifiers that categorize malware into different types, such as spyware, ransomware, and Trojans. The research highlights the effectiveness of a new Ensemble architecture, where a Deep Neural Network (DNN) model is combined with a logistic regression (LR) model compared to traditional Machine Learning (ML) algorithms, demonstrating statistically significant improvements in performance metrics. In conclusion, the study highlights the importance of innovative approaches in malware detection, especially in a context where obfuscation techniques are becoming increasingly sophisticated. The use of datasets such as CIC-MalMem-2022 and the application of techniques like Synthetic Minority Over-sampling TEchnique (SMOTE) to address data imbalances are crucial steps in advancing user privacy protection against cyber threats.

The article by Lazzarini et al. [12] proposes a method for intrusion detection using FL. For the development of the research, the authors utilize open-source tools and datasets, where discrete results were achieved. To validate the research, they used the ToN_IoT dataset and CICIDS2017, performing binary and multiclass classification with a shared artificial neural network (ANN) model and federated averaging (FedAvg) as the aggregation algorithm. To enrich the results, the authors evaluate three additional alternative aggregation methods, namely FedAvgM, FedAdam, and FedAdagrad, and compare their performances with FedAvg.

The article by Rajesh et al. [13] addresses two main challenges in intrusion detection in Industrial Internet of Things (IIoT) networks: the scarcity of specific data and the limitations of traditional Machine Learning algorithms in the face of high dimensionality and data heterogeneity. As a solution, the authors propose the use of Federated Transfer Learning (FTL) combined with a new convolutional neural network architecture, called Combo-NN, which integrates DNN and Convolutional Neural Network (CNN). The architecture is agnostic to the type of dataset, allowing the use of public IoT databases. The experiments demonstrated consistent performance between clients and the server, with minimal variations in accuracy across iterations, achieving up to 90% accuracy in the initial phase.

The SIM-FED article, [14] presents a malware detection model for IoT devices based on FL and Deep Learning, focusing on privacy preservation and security in distributed environments. The model utilizes a lightweight one-dimensional CNN with optimized hyperparameters, reducing the need for preprocessing and computational overhead. Among the evaluated aggregation strategies, FedAvg was adopted to consolidate local models. SIM-FED demonstrated robustness against white-box and black-box attacks, with minimal performance degradation, an aspect often overlooked in previous studies. The tests conducted with the IoT-23 dataset indicated superior performance compared to other approaches, achieving 99.52% accuracy. This approach is particularly relevant for distributed

and privacy-sensitive environments, such as IoT networks.

The article by Hossain et al. [15] proposes an advanced structure based on Machine Learning for the detection of obfuscated malware in memory dumps, considered one of the most challenging contexts in cybersecurity. The approach uses a comprehensive dataset, Obfuscated-MalMem2022, containing benign and malicious samples, including spyware, ransomware, and Trojans. The process includes rigorous preprocessing steps such as normalization, encoding, and class balancing through SMOTE. Feature selection is performed based on chi-squared tests, mutual information, and correlation analysis. The final model, based on ensemble learning techniques, outperformed previous approaches by achieving accuracy above 99% in both binary and multiclass classifications, demonstrating its effectiveness and robustness against sophisticated threats.

The article of Chirp et. al. [16] proposed the Deep-HMD, a deep learning-based model for zero-day malware detection using data collected from Hardware Performance Counters (HPCs), overcoming limitations of traditional software-based approaches. The HMD methods train standard ML algorithms on HPC events to develop accurate classifiers for detecting malicious software signatures. The proposal transforms hardware event data into images and applies a lightweight convolutional neural network architecture with residual blocks (ResNet) to improve detection. Nine types of malware were considered, with four used as zero-day threats (not seen during training). The model demonstrated superior performance compared to traditional ML classifiers, achieving a 97% F1-score, 98% accuracy, and 99% recall, confirming its effectiveness in detecting unknown malware in real-time.

The Wang et. al. [17] proposed hybrid approach that combines TL and deep learning to improve malware detection in executable files. The method involves converting PE (Portable Executable) files into byte images and using pretrained networks such as DenseNet to extract complex visual features. The model is fine-tuned with a small labeled dataset of malware, allowing for rapid adaptation to the new domain without significant loss of performance. The experiments showed that this technique outperformed traditional signature-based or heuristic methods, highlighting the robustness of TL in cybersecurity environments with emerging threats. In the experiments, the model achieved 97.3% accuracy, 96.8% precision, and 97.6% recall, surpassing traditional signature-based or heuristic methods. The results underscore the robustness of TL in detecting emerging threats in cybersecurity environments.

In a recent work [18], the authors presented a malware detection strategy based on TL, utilizing pre-trained models to extract relevant features from files converted into images. The study employs the EfficientNet architecture, which offers an excellent balance between performance and computational efficiency, adapted to classify PE files as benign or malicious. The technique demonstrated robustness against obfuscated samples, standing out for its ability to generalize to new variants of malware with minimal training. The approach reinforces the potential of TL as a scalable and effective alternative in modern information security systems. In the experiments, the model achieved an accuracy of 98.6%, a precision of 98.2%, and a recall of 98.9%, surpassing traditional methods based on signatures or heuristics. The approach reinforces the potential of TL as a scalable and effective alternative in modern information security systems.

Chen et al. [19], investigate the current challenges in intrusion detection in networks and propose the FETLSVM algorithm, which combines FL and TL to train models in a decentralized manner, with encrypted sharing of parameters, preserving data privacy. The approach is divided into three phases: model training in the cloud, local learning within organizations, and updating the global model. Experiments were conducted using the dataset CICIDS2017, and the results showed that FETLSVM outperformed baseline models (AdaBoost, GoogLeNet, AlertNet, and DeepFed) in accuracy, especially in complex attacks. For benign samples, FETLSVM achieved 98.89% accuracy; for Brute Force attacks, 97.23%; for XSS, 65.35%; and for SQL Injection, 22.85%—demonstrating significant gains in scenarios where conventional models performed much worse. These results validate the effectiveness and precision of the proposed model in distributed environments with different attack patterns.

The analyzed articles present solutions for network protection through Intrusion Detection Systems (IDS) or implement obfuscated malware detection techniques and evaluation of FL frameworks in isolation. The paper's proposal is to obtain results that demonstrate the use of FL frameworks in obfuscated malware detection as an important part of the process.

The works explore the use of ML to improve malware and intrusion detection in general, addressing aspects of FL techniques, classification techniques, and aggregation functions, highlighting the effectiveness and challenges of this approach in FL cybersecurity scenarios.

The differential of this article lies in its innovative approach to detecting malicious attacks on networks, utilizing FTL with a focus on preserving the privacy and security of institutional data. Additionally, it stands out for its dynamic scalability to include more organizations, providing adaptable flexibility to real scenarios. The concern for maintaining the generality of the model without loss of efficiency when transferred between different cores and organizations strengthens the robustness of the proposed solution, distinguishing it from other existing approaches.

III. Theoretical Foundation

Federated Learning was introduced in a Google paper in 2017 [20] and refers to the training of machine learning models in a decentralized and collaborative manner. This technique allows for data storage on edge devices, where models are trained and updates are sent to a central server. FL can be classified into three categories: horizontal federated learning, vertical federated learning, and federated transfer learning.

A. Horizontal Federated Learning

Horizontal Federated Learning (HFL), also known as sample-based Federated Learning, occurs when different devices or locations have the same features, but different instances [21].

In 2017, Google proposed a solution for HFL to update models on Android devices. In this model, each Android phone trains the model parameters locally and sends updates to a central server, allowing continuous improvement of the model without sharing raw data [22]. For example, regional banks may serve different customer groups but share the same financial variables. The work of Shokri et al. [23] proposes a collaborative deep learning scheme in which participants train models independently and share only subsets of updated parameters.

B. Vertical Federated Learning

Vertical Federated Learning (VFL), or attribute-based Federated Learning, applies when different organizations have data on the same users, but with distinct variables [24].

For instance, a bank and an e-commerce company may share a customer base in the same city. However, while the bank holds financial and credit data, the e-commerce company maintains browsing and purchase records. To build a predictive model that utilizes both sources without compromising privacy, VFL is applied.

Liu et al. [21] define VFL as an environment in which different parties collaboratively train machine learning models without exposing their raw data or model parameters.

C. Federated Transfer Learning

Federated Transfer Learning (FTL) is employed when the datasets differ in both samples and attributes. This method uses transfer learning to mitigate the scarcity of labeled data and the disparity between the attribute spaces of the datasets utilized [21].

Unlike HFL and VFL methods, which require some similarity in the data, FTL allows for the adaptation of models across distinct domains, enabling the reuse of acquired knowledge in one context to enhance a model in another.

D. Transfer Learning

Transfer learning (TL) is a widely used technique to optimize the training of Machine Learning (ML) models. This approach enables the reuse of a previously trained model on a dataset with characteristics similar to the new dataset, reducing the need to train the model from scratch [25].

ML models often require large volumes of data to achieve effective learning. TL mitigates this need by allowing previously trained models to be adapted to new problems with a smaller amount of data and in less time. This approach significantly reduces the computational cost and has been widely adopted by large technology companies [26].

The first reference to transfer learning in the context of neural networks was presented by Bozinovski and Fulgosi in 1976, in the paper The Influence of Pattern Similarity and Transfer of Learning upon Training of a Base Perceptron B2 [27]. Subsequently, in 1993, Pratt formulated the Discriminability-Based Transfer Algorithm (DBT), significantly contributing to the advancement of the technique [28].

E. Dataset

Datasets are structured collections of information used in the training and testing of models, being essential for the validation of experiments in Machine Learning and other data processing algorithms. In this study, we utilized datasets focused on Intrusion Detection Systems and Malware Detection, which contain samples of benign and malicious traffic.

Various types of datasets are available for different domains, such as computer vision, natural language processing, and data science. For this work, we employed a public dataset aimed at Malware Detection, suitable for the context of the study.

F. FedAvg (Weighted Average Algorithm)

The FedAvg method is the most widely used for implementing FL; its goal is to train models that can accurately predict output based on an input set, considering that the data for training the model is distributed across various edge devices. The main advantage of FedAvg is that it allows collaborative training without needing the samples stored on edge devices to be transmitted to the central server; data privacy and network overhead are two relevant points in this architecture.

By running two models from the same random initialization and training each one separately on different subsets of data, McMahan [20] observed that FedAvg produces surprisingly strong performance.

G. Federated Learning Frameworks

To facilitate the implementation of FL, there are frameworks that perform the necessary orchestration for using the technology. The frameworks are capable of managing the process of transferring the global model to edge devices, as well as the reverse flow, from the edge device to a central server. Furthermore, the framework should be responsible for the model aggregation activity, which must take place on the central server. Although the use of FL is quite recent, there are several Python libraries for developing applications. We can highlight some federated learning frameworks that stand out in the current landscape:

- Nvidia Flare; [29].
- TFF - TensorFlow Federated; [30]
- IBM FL Community Edition; [31].
- PySyft; [32].
- Flower – Friendly Federated Learning Framework. [33].

Different tools are analyzed in this study, with those that best fit the research objectives being selected.

H. DevOps

A set of practices that combines software development (Dev) and IT operations (Ops). DevOps aims to shorten the system development lifecycle while providing resources, fixes, and updates frequently in close alignment with business objectives. At the heart of the DevOps philosophy are the practices of Continuous Integration (CI) and Continuous Delivery (CD), which form the backbone of a successful DevOps implementation. Continuous Integration is a development practice where developers integrate code into a shared repository frequently,

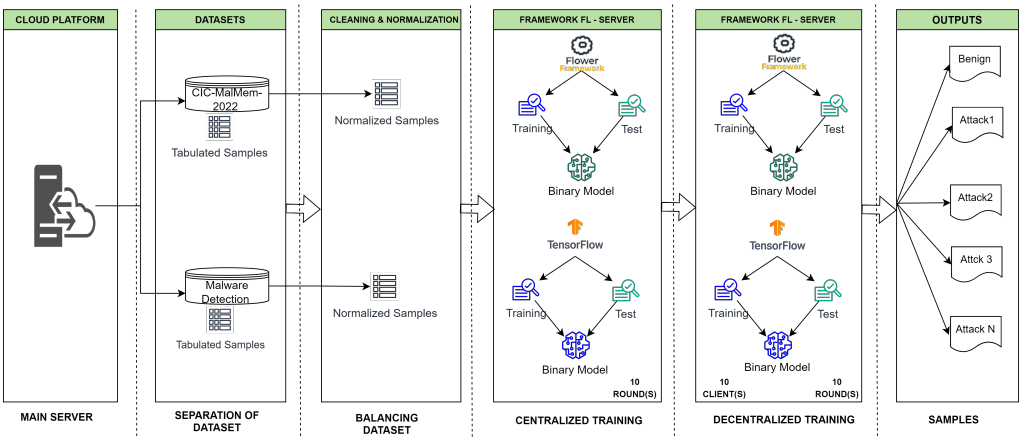


Fig. 1. Flow of the proposed model - The diagram illustrates the complete cycle of the model development process, spanning from sample balancing, centralized training, to decentralized execution with the Flower and TFF frameworks, culminating in the generation of the final results.

preferably several times a day. Each integration is verified by an automated build and automated tests to detect integration errors as quickly as possible. This practice helps ensure that the software is always in a state that can be released, significantly reducing the time needed to deliver new features and fixes [34].

IV. Methodology

The methodology describes the items that make up the project and are used during the experiment; it also defines the datasets to be used and the classifiers (models) that are most suitable. The datasets need to be trained using a parameterized and fine-tuned classifier so that this same classifier can subsequently train other datasets, initially with similar characteristics.

It is important to highlight that datasets were chosen for the work in a way that they can utilize the FL technique, focusing on characteristics such as user data privacy, decentralization, and collaboration. The following outlines the steps, techniques, and preparation for the tests and training that were carried out during the development of the project, according to the flow in Fig. 1.

A. Datasets

To achieve the desired results in testing and training, two data sets were selected that meet specific academic criteria, such as the inclusion of audit records, diversity of attacks, labeled normal traffic, and privacy assurance. Some characteristics were noted in the choice of the data sets:

- Includes audit logs and raw network data;
- Contains a variety of modern attacks;
- Represents realistic and diverse normal traffic;
- Is labeled;
- Provides privacy assurance;
- Is accepted by the community.

The selected data sets were Malware Memory Analysis CIC-MalMem-2022 [35] and Malware Detection Dataset [36].

B. Malware Memory Analysis CIC-MalMem-2022

The Malware Memory Analysis CIC-MalMem-2022[2] was created by the Canadian Institute for Cybersecurity. This data set is balanced, consisting of 50% malicious memory dumps and 50% benign, covering three categories: Trojan, Spyware, and Ransomware, containing 29.298 benign samples and 29.298 malicious samples [35].

C. Malware Detection Dataset

The Malware Detection Dataset[3], meticulously created in a Unix/Linux-based virtual environment, comprises 100.000 observations intended to classify Android applications into benign and malicious categories. Each observation encapsulates the behavior and characteristics of a distinct application across 35 diverse features, covering aspects such as system calls, network activity, and permission requests. This extensive set of features, combined with a balanced representation of harmless and malicious software, provides a comprehensive and granular perspective essential for the development of reliable Machine Learning models for malware detection [36].

Obfuscated malware is malware that conceals itself to avoid detection and eradication. The obfuscated malware dataset is designed to test detection methods for obfuscated malware through memory.

D. Data Preprocessing

The data extracted in its original form is not always ready for use in classifiers, requiring preprocessing to adjust and validate it. Data preprocessing is a data mining technique [37] that transforms raw data into a format suitable for analysis.

Real-world data often presents inconsistencies, incompleteness, and errors, as well as the absence of certain patterns. Preprocessing serves as an effective method to mitigate these problems [38].

[2]https://www.unb.ca/cic/datasets/andmal2020.html.
[3]https://www.kaggle.com/datasets/nsaravana/malware-detection.

In this study, data cleaning involves filling in missing values, smoothing noisy data, and resolving inconsistencies. Subsequently, data integration is performed, consolidating different representations and eliminating redundancies.

After this stage, the data is normalized and divided into training and testing sets. This practice, widely adopted in the literature, includes a separate validation set, used exclusively to evaluate the model's final performance. This work adopts a 70% split for training and 30% for testing, using the train-test-split strategy [39].

In decentralized training, the dataset is partitioned according to the number of clients, defining the size of each partition. The train-test-split function from the sklearn library is used to assist in this process. For the application of Federated Learning (FL) associated with Transfer Learning (TL), we use FedAvg as the aggregation function and models based on Local Neural Networks (LNN), chosen based on previous studies [40], [41].

The approach employed in this study includes FL with Recurrent Neural Networks (RNN) in a centralized format (FL server), and in a decentralized format (10 FL clients), we utilize LNN, TensorFlow Federated (TFF), and Flower with FedAvg to assess the model's effectiveness. Additionally, we incorporate DevOps techniques for knowledge transfer (TL).

Fig. 2 illustrates the application of Federated Transfer Learning (FTL), a paradigm where multiple organizations or devices collaborate in training a model without sharing private data. The diagram also represents the DevOps Pipeline, responsible for automatically creating, testing, and deploying a pre-trained model to perform TL.

At the top of the diagram, the central cloud represents the global server, which stores the initial model trained with an external dataset. This model is distributed to all institutions, which can refine it before using it internally. Each institution generates its own local model, tailored to its specific data.

Periodically, the parameters of the local models are aggregated on the central server using techniques such as FedAvg for updating the global model. After this update, the model is sent back to the devices or organizations for the next round of training.

In the proposed scenario, focused on the financial sector, each organization or device conducts training locally, keeping its data private and without the need to share it with the central server. In the corresponding diagram, the local models are represented as Model 1, Model 2, ..., Model k, illustrating the flow of learning between the participating institutions and the global server. The involved organizations — such as banks and financial institutions — contribute to the federated training through internal servers (D1, D2, ..., Dk), without exposing sensitive information. Edge devices, such as laptops and smartphones, also integrate into the computational process, enabling collaborative learning while preserving privacy.

However, the heterogeneity among different branches of activity can directly impact the performance of the federated model, influencing aspects such as: (i) the variability of local data, which complicates the convergence of the global model; (ii) the effectiveness of knowledge transfer in Transfer Learning-based architectures, especially when the feature space is not shared among domains; and (iii) the capacity of the model to generalize in environments with low statistical similarity between the data (non-IID).

As illustrated in Fig. 2, training in FL occurs at the level of edge devices or organizations, while TL leverages a pre-trained model within a DevOps pipeline. This model reduces training time, the need for large volumes of data, and improves generalization. For the development of the models, we use two Federated Learning frameworks, evaluating criteria such as client-side training capability, server-side aggregation, and communication efficiency. Additionally, we check whether the frameworks support simulation, ensuring the reproducibility of experiments.

E. Flower Framework

The Flower Framework [33] was originated as a research project at the University of Oxford and is distributed by the German company Adap GmbH. Developed as a flexible and open-source framework for Federated Learning (FL), it stands out for its ease of use, comprehensive documentation, and independent configuration, allowing machine learning (ML) models to be easily adapted to the federated setting.

Flower enables customization, extension, or replacement of components, making it suitable for new aggregation algorithms and research applications. Additionally, it features interoperability with different operating systems, support for a wide variety of devices, and scalability for millions of clients. However, it does not include native functionalities for privacy, monitoring, deployment, or orchestration.

In this study, Flower is applied to the CIC-MalMem-2022 and Malware Detection Database datasets after data preprocessing. LNN is then used to train the proposed model. The training and evaluation of the ML model are carried out using PyTorch, considering metrics such as loss, accuracy (ACC), area under the curve (AUC), recall, and precision, for both the training and test data over 10 epochs.

F. TensorFlow Federated (TFF)

TensorFlow Federated (TFF) [30] is an open-source framework for FL, developed by Google, initially focused on mobile keyboard predictions and on-device research. TFF is actively used by Google, allowing for the adaptation of TensorFlow/Keras models to federated settings, as well as providing support for TensorBoard and enabling the implementation of custom aggregation algorithms.

To simulate a FL scenario, the training data (X_{train} and y_{train}) and test data (X_{test} and y_{test}) are equally divided among NUM_CLIENTS clients and stored in a list called *client_data*. Each client receives 4.354 samples, except for three clients that receive one extra piece of data, resulting in the following distribution: clients 1 to 3 have 4.355 samples, while clients 4 to 10 have 4.354 samples in the CIC-MalMem-2022 dataset. In the Malware Detection Dataset, each client receives 7.500 samples.

In this study, TFF is applied with the same configurations and proportions as Flower, both on the server and on the clients, ensuring compatibility between the two approaches.

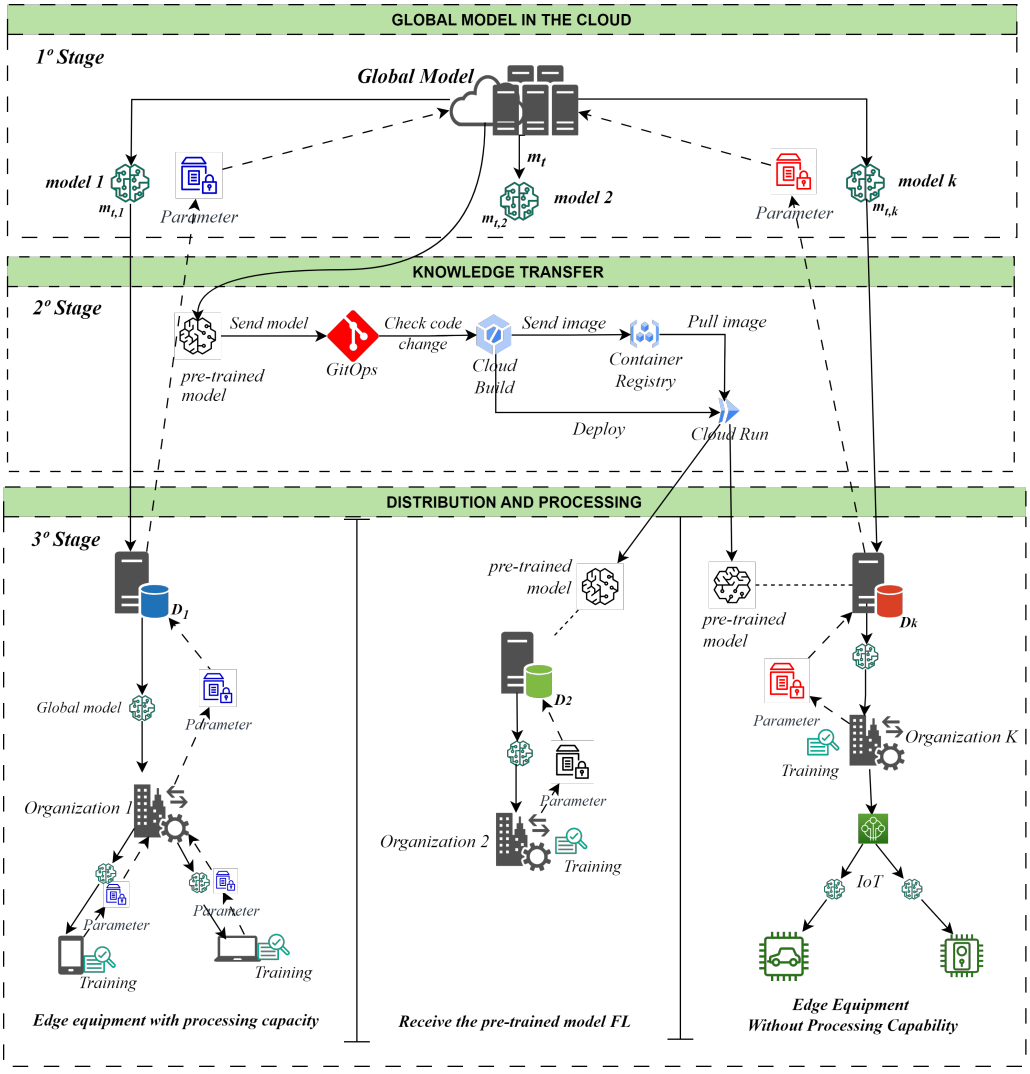


Fig. 2. Federated Transfer Learning Diagram - The figure illustrates the FTL architecture, structured into three main stages: the first shows the global model stored in the cloud and the process of parameter exchange; the second describes the DevOps Pipeline responsible for the automated deployment of the pre-trained model; and the third highlights two decentralized processing scenarios using FL, demonstrating the interactions between clients and the central server.

For the execution of the experimental tests, the following frameworks and libraries were used: Python 3.10, TensorFlow 2.14, PyTorch 2.1, Flower 1.5.0, and TensorFlow Federated (TFF) 0.61. The experiments were conducted in a computational environment with the following hardware: Intel® Core™ i7-1255U processor (10 cores, 12MB of cache, frequency up to 4.7GHz), Windows 11 Home 64-bit operating system (in Portuguese), integrated Intel® Iris® Xe GPU with shared graphics memory, and 16GB of DDR4 RAM (2x8GB).

V. Results

The results obtained in the tests demonstrate excellent performance on the CIC-MalMem-2022 and Malware Detection datasets, using TFF and Flower. For a more detailed analysis, the metrics ACC, Precision, and Recall are evaluated, and the confusion matrix for each model (centralized and decentralized) from the conducted tests is also generated. Furthermore, the training time for each model is analyzed.

A. Results in Centralized Tests

The results obtained from the CIC-MalMem-2022 dataset using TFF and RNN indicate that the ACC progressively increases, stabilizing at 0.9992 starting from the 6th epoch. The metrics Precision, Recall, and AUC reach values of 0.9994, 0.9992, and 0.9992, respectively, in the tests conducted on the FL server. The consistent trajectory of these metrics signals stability and convergence of the model. Fig. 3 illustrates the values of the ACC, Precision, and Recall over the epochs.

The tests conducted with the Malware Detection dataset and TFF, as shown in Fig. 3, second graph, show that the ACC starts at 0.9443 and reaches its best index of 0.9997 in the 10th epoch, indicating that the model efficiently captures the patterns of the test data, demonstrating an excellent ability to generalize.

The AUC rises from 0.9442 to 0.9998, indicating perfect separation between the classes.

The model exhibits excellent metrics in both Precision and

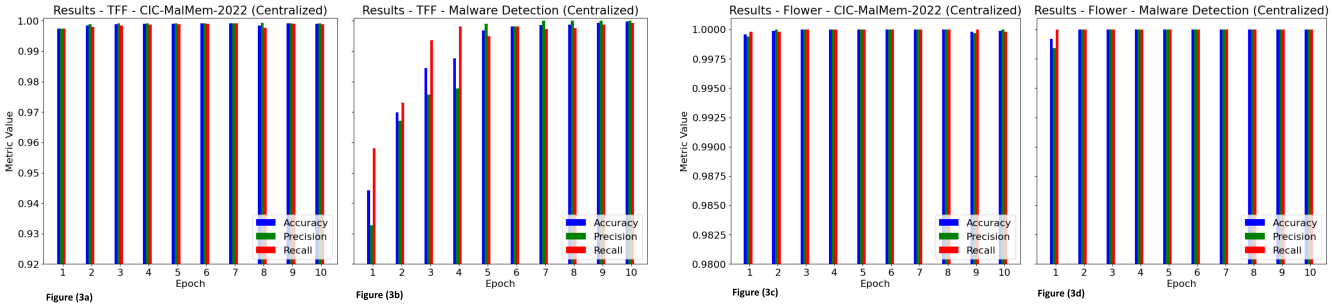


Fig. 3. Centralized Diagram - Test result Accuracy, Precision and Recall.- Figure (3a): TFF with CIC-MalMem-2022 dataset (centralized). - Figure (3b): TFF with Malware Detection dataset (centralized). - Figure (3c): Flower with CIC-MalMem-2022 dataset (centralized). - Figure (3d): Flower with Malware Detection dataset (centralized). Each subfigure presents Accuracy (blue), Precision (green), and Recall (red) metrics.

TABLE I
COMPARATIVE RESULTS BETWEEN TFF AND FLOWER - TEST - CENTRALIZED

Dataset	ACC Initial	Best ACC	AUC	Loss	Precision	Recall	Time	Framework	Samples
CIC-Malware	0.9975	0.9992 (6ª época)	0.9992	0.0018	0.9994	0.9992	**36s**	TFF	13.064
Malware Detection	0.9443	0.9997 (10ª época)	0.9998	0.0016	**1.0**	0.9994	68s	TFF	22.500
CIC-Malware	**0.9996**	1.0 (3ª época)	**1.0**	0.0014	**1.0**	**1.0**	**23s**	Flower	13.064
Malware Detection	0.9992	**1.0 (2ª época)**	**1.0**	**0.0002**	**1.0**	**1.0**	46s	Flower	22.500

Recall during training and testing, with AUC close to 1.0. The testing and training time of TFF in Malware Detection is 68 seconds, a relatively optimal value considering the larger number of records and the lower dimensionality of the dataset.

The tests with the Flower Framework on the CIC-MalMem-2022 dataset Fig. 3, third graph, show consistent and high ACC, starting at 0.9996 and reaching 1.0 in the 3rd epoch, with a slight decrease to 0.9998 in the 9th epoch. This drop may be associated with fluctuations in the test data.

The Precision rises from 0.9994 in the 1st epoch to 1.0 in the 2nd epoch, with a partial drop to 0.9997 in the 9th, ending at 1.0 in the 10th epoch. The Recall maintains values of 0.9998 for the 1st and 2nd epochs, then reaches 1.0 for the remaining epochs, except in the 10th epoch where it returns to 0.9998. The training time in the Flower Framework is 23 seconds, demonstrating efficiency.

In the tests conducted on the Malware Detection dataset with Flower, the ACC starts at 0.9992 in the 1st epoch, reaching 1.0 in the 2nd epoch and maintaining that value until the 10th epoch. The Precision follows the same trend but starts at 0.9984 in the 1st epoch. The Recall maintains an index of 1.0 for all epochs. Despite the excellent results, constant metrics may indicate that the model does not learn significantly after the initial epochs. The training time of Flower is 46 seconds, shorter than the time for the same model trained with TFF.

The Table I summarizes the results of the tests conducted on the centralized model application.

In Table II, the results of the centralized tests for the TFF and Flower models are presented. It is noted that the Flower model stands out significantly, exhibiting superior performance across the two analyzed datasets. Notably, the values for False Positives (FP) and False Negatives (FN) reached minimal levels, going down to zero, which highlights the model's effectiveness in correctly classifying the samples. This performance suggests greater accuracy and reliability of Flower compared to TFF in detecting the analyzed instances.

B. Results in Decentralized Testing

In the experiments with decentralized FL, the average values of the test metrics ACC, Precision, Recall, and AUC are also extracted over the epochs.

The decentralized model trained with TFF on the CIC-MalMem-2022 dataset, using 10 clients with approximately 1,306 records per client, achieves good values of ACC, starting at 0.9992 in the first epoch and reaching its best value of 0.9998.

The metrics for Precision, Recall, and AUC reach values of 1.0, 0.9997, and 1.0, respectively, in a time of 36 seconds. Notably, AUC demonstrated 1.0 in all epochs. The results for the metrics of ACC, Precision, and Recall can be observed in the first graph in Fig. 4.

In the Fig. 4, the third diagram, illustrates the results of the Flower Framework applied to the CIC-MalMem-2022 dataset in the decentralized test. The ACC applied to the test set starts at 0.9996, reaching 1.0 in the 2nd epoch. The AUC follows the same pattern, beginning at 0.9977 and achieving 1.0 in the 4th epoch. The model execution time is approximately 128 seconds per client; despite the time being longer than TFF with the same dataset, it presents better results. This can be observed in the FP and FN in Table IV.

Lastly, Fig. 4, the fourth diagram, illustrates the results of the Flower Framework applied to the Malware Detection dataset in the decentralized test, where the tests for ACC start at 1.0 in the 1st epoch and remain at that value until the end.

TABLE II
CONFUSION MATRIX - COMPARISON BETWEEN TFF AND FLOWER - CENTRALIZED TEST

	TFF - Centralized Test								Flower - Centralized Test							
	TFF + CIC-Malware				TFF + Malware Detection				Flower + CIC-Malware				Flower + Malware Detection			
Epoch	**TN**	**FP**	**FN**	**TP**	**TN**	**FP**	**FN**	**TP**	**TN**	**FP**	**FN**	**TP**	**TN**	**FP**	**FN**	**TP**
1	6578	17	16	6453	34889	8	7	35096	6591	4	1	6468	11185	18	0	11297
2	6588	7	13	6556	34895	2	4	35099	6595	0	1	6468	11203	**0**	**0**	11297
3	6590	5	10	6459	34894	3	1	35102	6595	**0**	**0**	6469	11203	**0**	**0**	11297
4	6590	5	8	6461	34895	2	1	35102	6595	**0**	**0**	6469	11203	**0**	**0**	11297
5	6590	5	7	6462	34895	2	0	35103	6595	**0**	**0**	6469	11203	**0**	**0**	11297
6	6590	5	6	6463	34897	**0**	**0**	35103	6595	**0**	**0**	6469	11203	**0**	**0**	11297
7	6590	**5**	**5**	6464	34896	1	1	35102	6595	**0**	**0**	6469	11203	**0**	**0**	11297
8	6591	4	15	6454	34897	0	1	35102	6595	**0**	**0**	6469	11203	**0**	**0**	11297
9	6590	5	6	6463	34897	**0**	**0**	35103	6595	**0**	**0**	6469	11203	**0**	**0**	11297
10	6590	5	7	6462	34897	**0**	**0**	35103	6595	**0**	**0**	6469	11203	**0**	**0**	11297

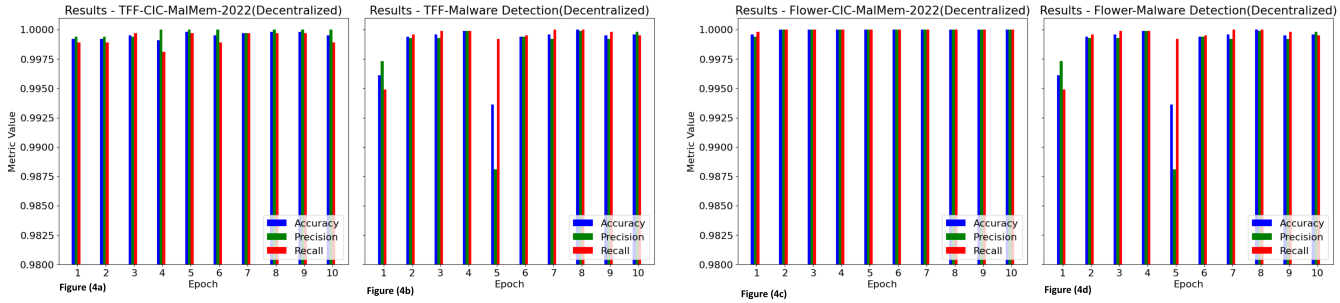


Fig. 4. Decentralized Diagram - Test result Accuracy, Precision and Recall - Figure (4a): TFF with CIC-MalMem-2022 dataset (Decentralized). - Figure (4b): TFF with Malware Detection dataset (Decentralized). - Figure (4c): Flower with CIC-MalMem-2022 dataset (Decentralized). - Figure (4d): Flower with Malware Detection dataset (Decentralized). Each subfigure presents Accuracy (blue), Precision (green), and Recall (red) metrics.

TABLE III
COMPARATIVE RESULTS BETWEEN TFF AND FLOWER - DECENTRALIZED TEST

Dataset	**ACC Initial**	**Best ACC**	**AUC**	**Loss**	**Precision**	**Recall**	**Time**	**Framework**	**Samples**
CIC-Malware	0.9992	0.9998 - 5ª época	**1.0**	0.0035	**1.0**	0.9997	**36s**	TFF	1.306
Malware Detection	0.9961	1.0 - 8ª época	**1.0**	0.0014	0.9999	**1.0**	86s	TFF	2.250
CIC-Malware	0.9996	1.0 - 2ª época	**1.0**	0.0002	**1.0**	**1.0**	128s	Flower	1.306
Malware Detection	**1.0**	**1.0 - 1ª época**	**1.0**	**0**	**1.0**	**1.0**	**46s**	Flower	2.250

In the tests, AUC starts at 0.9950 and reaches 1.0 in the 3rd epoch, remaining constant until the last epoch. The training time is 46 seconds.

Goodfellow et al. [42] highlight that high and consistent ACC with a low and consistent loss during training and testing are indicative that the model does not show signs of problems. This behavior may occur due to the ease of classifying the data or the high suitability of the model for the task.

However, it is observed that the comparative results between TFF and the centralized Flower test are presented in Table 1; however, the most relevant results are not described or discussed, which contributes to a better understanding of the results.

The same occurs with the presentation of results in Table III. Table I shows that both frameworks achieve excellent performances in ACC, AUC, Precision, and Recall, close to or equal to 1.0; however, Flower achieves the best results in fewer epochs and with less time. In the CIC-Malware, Flower reaches an ACC of 1.0 by the 3rd epoch, compared to the 6th epoch for TFF.

In Malware Detection, Flower achieves 1.0 by the 2nd epoch, while TFF requires 10 epochs. It is also noted that Flower is more efficient in execution time, consuming fewer seconds per experiment, which can be relevant in contexts with computational constraints. In Table III, it is observed that TFF and Flower demonstrate almost perfect performance, with ACC, AUC, Precision, and Recall metrics close to or equal to 1.0. The Loss values in Flower are lower, indicating less classification error during testing.

Regarding speed and convergence efficiency, Flower is superior to TFF, achieving an ACC of 1.0 by the 1st epoch for the Malware Detection dataset, and by the 2nd epoch for CIC-Malware, while TFF requires 5 to 8 epochs to achieve similar performance. This demonstrates a greater learning

TABLE IV
CONFUSION MATRIX - COMPARISON BETWEEN TFF AND FLOWER - DECENTRALIZED TEST

	TFF - Decentralized Test								**Flower - Decentralized Test**							
	TFF + CIC-Malware				**TFF + Malware Detection**				**Flower + CIC-Malware**				**Flower + Malware Detection**			
Epoch	**TN**	**FP**	**FN**	**TP**	**TN**	**FP**	**FN**	**TP**	**TN**	**FP**	**FN**	**TP**	**TN**	**FP**	**FN**	**TP**
1	6591	4	7	6462	10422	781	473	10824	6591	4	1	6468	11203	**0**	**0**	11297
2	6591	4	7	6462	10829	374	304	10993	6595	0	1	6468	11203	**0**	**0**	11297
3	6591	4	2	6467	10923	280	72	11225	6595	0	42	6427	11203	**0**	**0**	11297
4	6595	0	12	6457	10947	256	21	11276	6595	**0**	**0**	6469	11203	**0**	**0**	11297
5	6595	**0**	**2**	6467	11192	11	58	11239	6595	**0**	**0**	6469	11203	**0**	**0**	11297
6	6595	0	7	6462	11183	20	21	11276	6595	**0**	**0**	6469	11203	**0**	**0**	11297
7	6593	2	2	6467	11203	0	31	11266	6595	**0**	**0**	6469	11203	**0**	**0**	11297
8	6595	**0**	**2**	6467	11203	0	27	11270	6595	**0**	**0**	6469	11203	**0**	**0**	11297
9	6595	**0**	**2**	6467	11203	0	13	11284	6595	**0**	**0**	6469	11203	**0**	**0**	11297
10	6595	0	7	6462	11203	0	7	11290	6595	**0**	**0**	6469	11203	**0**	**0**	11297

efficiency of Flower in the decentralized environment. In terms of execution time, Flower in CIC-Malware is greater (128s compared to 36s for TFF), suggesting that the efficiency of ACC comes at a higher computational cost in this case. For the Malware Detection dataset, Flower is faster (46s) than TFF (86s), showing better performance in both quality and time in this scenario. Another positive point for Flower is that despite the sample sizes, it maintains its performance superior to or equal to TFF.

Comparative Analysis between TFF and Flower

Performance in Centralized Scenario (Table I):

- Final Accuracy: Both frameworks achieve excellence (ACC $\geq$ 0.9992); however, Flower reaches 1.0 on both datasets, slightly surpassing TFF (0.9992 to 0.9998).
- Convergence Speed: Flower shows a significant advantage, achieving the best accuracy already in the 2nd and 3rd epochs, while TFF requires the 6th and 10th epochs.
- Computational Efficiency: Flower exhibits a shorter execution time compared to TFF (23s vs. 36s) on the CIC-Malware dataset and Malware Detection (46s vs. 68s).

Performance in Decentralized Scenario (Table III):

- Stability: Flower maintains perfect performance (ACC $\geq$ 0.9996 across all epochs), while the TFF reaches ACC ($\geq$ 0.9992), indicating greater sensitivity to the data distribution.
- Communication Overhead: Flower's processing time increases drastically in decentralized environments using the CIC-Malware, while TFF maintains stable times. This suggests that TFF may be more suitable for networks with latency constraints. Although both frameworks achieve high ACC ($\geq$ 0.9992), the choice between TFF and Flower depends on the scenario:
- Priority on maximum ACC: Flower is ideal, especially in centralized environments.
- Applications sensitive to latency: TFF offers a better balance between time and performance in decentralized systems when applied to the CIC-Malware dataset, which contains a smaller number of samples.

Flower's perfect results (Table III) require caution; we suggest future analyses with unbalanced datasets to assess robustness.

This can also be observed in the confusion matrix of the table IV, where the values of FP and FN remain at zero since the first epoch.

VI. CONCLUSION

The analysis of the results obtained with the FL models, using the CIC-MalMem-2022 and Malware Detection datasets, confirms an excellent performance, both in centralized and decentralized configurations. The metrics (ACC), (Precision), (Recall), and (AUC) are close to 1.0, combined with the results from the confusion matrix, reflecting the models' ability to learn and generalize efficiently, as well as their operational efficiency in terms of training time.

The Flower framework demonstrates superior performance in both centralized and decentralized scenarios, while the TFF framework excels in execution time in tests conducted with the CIC-MalMem-2022 dataset using the decentralized model. However, it has a longer execution time when applied to Malware Detection, which involves a larger amount of data. These findings emphasize the importance of selecting the most suitable approach for the specific needs of the project, indicating that both TFF and Flower play crucial roles in the evolution of malware detection solutions.

The model applied to the detection and mitigation of obfuscated malware demonstrates high efficiency on the utilized datasets, validating the effectiveness of FL in this area. Despite some limitations, such as the need for minimum infrastructure, use of simulated data, presence of non-IID data, and absence of tests against adversarial attacks, the results are promising. The proposal shows potential for application in real environments that require high security and privacy preservation, such as hospitals, financial institutions, and research centers.

These results reinforce the need for continuous evolution and optimization of learning techniques to strengthen cybersecurity, which is essential in an increasingly digital world. In this way, we encourage the academic community and professionals in the field to deepen their studies and investigations, further exploring the potential of FL in combating threats and protecting data.

For future research, it is suggested to explore alternative aggregation methods such as FedAvgM, FedAdam, and

FedAdagrad, as well as to enhance the FTL technique, since there are no FL frameworks with this native functionality. It is also proposed to conduct studies with eXplainable AI (XAI) approaches, such as federated SHAP or federated attribution maps, focused on FL. These areas remain underexplored in academia, presenting promising opportunities for the development of more efficient models for IoT devices and edge devices. We also propose to mitigate vulnerabilities when malicious clients attempt to inject modified updates with the aim of weakening the model using false positive samples. Another proposal is the creation of a probabilistic mechanism to classify malware by offensive performance level, assisting in the assessment of the severity of threats and in prioritizing defensive responses in distributed environments.

ACKNOWLEDGMENTS

The authors are grateful to Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Brazil grants #2023/12830-0, and Petrobras, Brazil grant 2023/00466-1 for their financial support.

The authors are grateful to State Center for Technological Education “Paula Souza” (CEETEPS) for research conditions at the Defensive Cybersecurity and Artificial Intelligence Laboratory .

REFERENCES

[1] T. Carrier, P. Victor, A. Tekeoglu, and A. H. Lashkari,”Detecting obfuscated malware using memory feature engineering,” *in Proc. 8th Int. Conf. Inf. Syst. Security Privacy (ICISSP)*, 2022, pp. 177–188. DOI: 10.5220/0010770300003115.

[2] H. Huseynov, K. Kourai, T. Saadawi, and O. Igbe, ”Virtual machine introspection for anomaly-based keylogger detection,” *in Proc. IEEE 21st Int. Conf. High Perform. Switching Routing (HPSR)*, May 2020, pp. 1–6. DOI: 10.1109/HPSR48589.2020.9098970.

[3] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, ”Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence,” *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 2, pp. 341–351, Apr. 2020. DOI: 10.1109/TETC.2017.2768038.

[4] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, ”Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” *IEEE Transl. J. Magn. Japan*, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982]. DOI: 10.1109/TJMJ.1987.4563869.

[5] M. Dener, G. Ok, and A. Orman, ”Malware detection using memory analysis data in big data environment,” Appl. Sci., vol. 12, no. 17, p. 8604, Aug. 2022. DOI: 10.3390/app12178604.

[6] C.-W. Chen, C.-H. Su, K.-W. Lee, and P.-H. Bair, ”Malware family classification using active learning by learning,” *in Proc. 22nd Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2020, pp. 590–595. DOI: 10.23919/ICACT48636.2020.9061480.

[7] Kairouz, Peter et al. Advances and open problems in federated learning. Foundations and trends® in machine learning, v. 14, n. 1–2, p. 1-210, 2021. https://doi.org/10.48550/arXiv.1912.04977

[8] Estudos, R. T. d. I. d. S. n. B. Centro de. Incidentes notificados ao cert. br, 2025.

[9] Symantec (Broadcom). (2022). The Ransomware Threat Landscape: What to Expect in 2022. Disponível em: https://www.symantec.broadcom.com/

[10] S. P. Karimireddy, et al., ”Federated Learning Showdown: The Comparative Analysis of Federated Learning Frameworks,” 2023 *Eighth International Conference on Fog and Mobile Edge Computing (FMEC)*, 2023. DOI: 10.1109/FMEC57357.2023.00012.

[11] D. Cevallos-Salas, et al., ”Obfuscated Privacy Malware Classifiers based on Memory Dumping Analysis,” IEEE Access, 2024. DOI: 10.1109/ACCESS.2024.3045678.

[12] R. Lazzarini, H. T. Tianfield, and V. Charissis, ”Federated learning for IoT intrusion detection,” AI, vol. 4, no. 3, pp. 509-530, 2023. DOI: 10.3390/ai4030032.

[13] Rajesh, Lochana Telugu et al. Give and take: Federated transfer learning for industrial iot network intrusion detection. In: 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2023. p. 2365-2371. https://doi.org/10.48550/arXiv.2310.07354

[14] Nobakht, M., Javidan, R., & Pourebrahimi, A. (2024). SIMFED: Secure IoT Malware Detection Model with Federated Learning. Computers and Electrical Engineering, 109139. https://doi.org/10.1016/j.compeleceng.2024.109139

[15] Hossain, Md Alamgir; ISLAM, Md Saiful. Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity. Cybersecurity, v. 7, n. 1, p. 16, 2024. https://doi.org/10.1186/s42400-024-00205-z

[16] HE, Xinyue et al. Deep-HMD: A Deep Learning-based Framework for Accurate and Efficient Hardware-based Zero-day Malware Detection. Computers & Security, 2022. DOI: 10.1016/j.cose.2022.102869

[17] Wang, S., Zhang, Y., & Chen, J. (2022). Malware Detection Using Deep Transfer Learning Based on Bytecode Images. 2022 IEEE International Conference on Intelligence and Security Informatics (ISI), 1–6. https://doi.org/10.1109/ISI55183.2022.00013

[18] Al-Dujaili, A., Huang, Y., Hemberg, E., & O’Reilly, U. M. (2021). Adversarial deep learning for robust detection of binary encoded malware. Machine Learning, 110, 251–279. https://doi.org/10.1007/s10994-020-05909-6

[19] Chen, S. et al. An efficient intrusion detection method based on federated transfer learning. In: IEEE. 2024 3rd International Conference on Artificial Intelligence and Computer Information Technology (AICIT). [S.l.], 2024. p. 1–4. DOI: 10.1109/AICIT62434.2024.10730009.

[20] McMahan, Brendan, et al. ”Communication-efficient learning of deep networks from decentralized data,” *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, 2017. DOI: 10.48550/arXiv.1602.05629.

[21] Liu, Yang, et al. ”Vertical federated learning: Concepts, advances, and challenges,” *IEEE Transactions on Knowledge and Data Engineering* (2024). DOI: 10.1109/TKDE.2022.3141672.

[22] K. Bonawitz, et al., ”Practical secure aggregation for privacy-preserving machine learning,” in Proc. DOI: 10.1145/3133956.3133982

[23] Shokri, Reza, and Vitaly Shmatikov. ”Privacy-preserving deep learning,” *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015. DOI: 10.1145/2810103.2813687.

[24] S. Yang, et al., ”Parallel distributed logistic regression for vertical federated learning without third-party coordinator,” arXiv preprint arXiv:1911.09824, 2019. DOI 10.48550/arXiv.1911.09824.

[25] Zhang, Yongxuan, and Jun Yan. ”Domain-adversarial transfer learning for robust intrusion detection in the smart grid.” 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). IEEE, 2019. DOI: 10.1109/SmartGridComm.2019.8909782.

[26] Wu, Peilun, Hui Guo, and Richard Buckland. ”A transfer learning approach for network intrusion detection,” *2019 IEEE 4th International Conference on Big Data Analytics (ICBDA). IEEE*, 2019. DOI: 10.1109/ICBDA.2019.8713205.

[27] Bozinovski, Stevo; Fulgosi, Ante. The influence of pattern similarity and transfer learning upon training of a base perceptron b2. In: Proceedings of symposium informatica. 1976. p. 121-126. https://doi.org/10.31449/inf.v44i3.2828

[28] Pratt, Daniel D. ”Andragogy after twenty-five years,” *New directions for adult and continuing education* 57.57 (1993): 15-23. DOI:10.1002/ace.36719935704

[29] NVIDIA, ”NVFlare: NVIDIA Federated Learning Application Runtime Environment,” [Online]. Available: https://nvflare.readthedocs.io. [Accessed: 2024-12-01].

[30] TensorFlow Federated Authors, ”TensorFlow Federated: Machine learning on decentralized data,” [Online]. Available: https://www.tensorflow.org/federated. [Accessed: 2024-12-02].

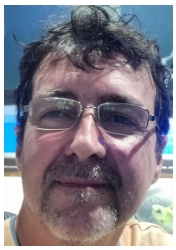
[31] IBM Research, ”IBM Federated Learning Community Edition,” [Online]. Available: https://ibmfl.mybluemix.net. [Accessed: 2024-12-04].

[32] OpenMined, ”PySyft: A Library for Secure and Private Machine Learning,” [Online]. Available: https://github.com/OpenMined/PySyft. [Accessed: 2024-11-13].

[33] Flower Authors, ”Flower: A Friendly Federated Learning Framework,” [Online]. Available: https://flower.dev. [Accessed: 2024-12-02]. DOI: 10.48550/arXiv.2007.14390.

[34] Banala, Subash. "DevOps Essentials: Key Practices for Continuous Integration and Continuous Delivery," *International Numeric Journal of Machine Learning and Robots* 8.8 (2024): 1-14.
[35] Canadian Centre for Cyber Security (CCCS) and Canadian Institute for Cybersecurity (CIC), "CCCS-CIC-AndMal-2020 Dataset," 2020. [Online]. Available: https://www.unb.ca/cic/datasets/andmal2020.html. [Accessed: 2024-09-21].
[36] N Saravana. "Detecção de malware—Kaggle," 2018. [Online]. Available: https://www.kaggle.com/datasets/nsaravana/malware-detection
[37] Mining, What Is Data. Data mining: Concepts and techniques. Morgan Kaufinann, v. 10, n. 559-569, p. 4, 2006.
[38] García, Salvador et al. Data preprocessing in data mining. Cham, Switzerland: Springer International Publishing, 2015.
[39] J. Tan, et al., "A critical look at the current train/test split in machine learning," arXiv preprint arXiv:2106.04525, 2021. DOI: 10.48550/arXiv.2106.04525
[40] V. Gazeau, K. Gupta, and M. K. An, "Advancements of Machine Learning in Malware and Intrusion Detections". 2024 *International Conference on Computer, Information and Telecommunication Systems (CITS)*, Girona, Spain, 2024, pp. 1-7. DOI: 10.1109/CITS61189.2024.10608018.
[41] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," IEEE Access, vol. 7, pp. 46717-46738, 2019. DOI: 10.1109/ACCESS.2019.2906934.
[42] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Carlos de Jesus Reis holds a bachelor's degree in Information Technology (2003), a specialisation in Business Administration from Mackenzie Presbyterian University (2008), and a master's degree in Computer Science from São Paulo State University (2025). He has worked for over 20 years in the financial and healthcare sectors, with stints at institutions such as [B]³ – Brazilian Stock Exchange. His main research interests include embedded systems, intrusion detection using machine learning, and agile deployment processes, with an emphasis on practices such as DevSecOps.



Carlos Alexandre Carvalho Tojeiro holds a B.Sc. in Systems Analysis and Information Technology from the College of Technology of Ourinhos (2008), an MBA in Business Management from the Faculty of Higher Education of Santa Bárbara (2011), and specializations in Higher Education from Estácio de Sá College (2015) and in Computer Network Security from the College of Technology of Ourinhos (2016). He received his M.Sc. in Computer Science from São Paulo State University (UNESP) in 2024. He is currently a Professor at the São Paulo State Centre for Technological Education. Carlos is the author of 11 papers published in indexed journals. His research interests include computer network security and applied cybersecurity practices.

Thiago José Lucas holds a B.Sc. in Information Security from the College of Technology of Ourinhos, an M.Sc. in Computer Science from the Institute of Biosciences, Letters and Exact Sciences at São Paulo State University "Júlio de Mesquita Filho" (UNESP – São José do Rio Preto), and a Ph.D. in Computer Science from the Advanced Network Security Laboratory at UNESP (LARS/UNESP – Bauru). He was affiliated with the Federal University of Technology – Paraná (UTFPR), where he specialized in the design and implementation of computer systems. His research interests include computer security, intrusion detection, and secure software development.

Kelton A. P. da Costa holds a B.Sc. in Systems Analysis from the Sacred Heart University (USC), an M.Sc. in Computer Science from Eurípedes University of Marília (UNIVEM), and a Ph.D. in Computer Science from the University of São Paulo (USP). He completed postdoctoral research in Computer Networks at the University of Campinas (UNICAMP) and in Anomaly Detection in Computer Networks at São Paulo State University (UNESP). He is currently a Professor in the Computer Science and Information Systems programs at São Paulo State University "Júlio de Mesquita Filho" (UNESP – Bauru campus), and a permanent faculty member in the M.Sc. and Ph.D. programs in Computer Science at UNESP (Bauru). His primary research interests include Computer Network Management, Cybersecurity, Anomaly Detection Systems, Network Signatures, and Quantum Machine Learning.