4

Federated Learning for Malware Detection in Edge devices

Dimitrios Serpanos and Georgios Xenos

University of Patras, Greece

Abstract

Malware detection is fundamental to safe and secure computing systems, from the cloud to Internet of Things (IoT) devices and Operational Technology (OT) systems. Malware detection is a process that inputs software samples, extracts their static and dynamic features and classifies them as malware or benign exploiting a range of Machine Learning (ML) algorithms and Deep Neural Networks (DNNs). The need for significant amounts of training data to obtain effective and efficient detection models is limited by the absence of sufficient benchmark datasets and by the intellectual property and privacy constraints that do not allow for data sharing among organizations.

In our work, we present an effective Federated Learning (FL) solution for malware detection, which achieves high accuracy in malware detection with a detection model that is developed in a distributed fashion among members of a federation that are not required to exchange source data. We consider a federation of Edge or near-Edge devices that are deployed as security providers for their organizational networks. Each device trains its own neural network (NN) model with its own data; local models are combined in a global, aggregated NN model exploiting cross-silo FL, and the global model is distributed to the federation members. We evaluate the FL solution with the EMBER dataset and demonstrate that our approach reaches accuracy above 93%, which is the accuracy of the non-federated centralized NN model. Our work demonstrates that our FL solution is effective and efficient achieving high accuracy without need to exchange source data, i.e. respecting privacy,

while it scales well with the size of the federation. Importantly, the approach demonstrates that organizations are highly motivated to participate in the federation because they achieve significantly higher malware detection accuracy than the one they would achieve by exploiting only their own training data.

Keywords: federated learning, malware detection, deep learning.

4.1 Introduction and Background

Malware detection is a process where software samples are analysed, features are extracted from them and classified -as malware or benign- based on the extracted features. The typical process is shown in Figure 4.1 which presents the operational structure of Sisyfos [1], a representative malware and analysis platform which we use as our target pilot platform. For analysis of a sample, Sisyfos processes the sample and extracts two categories of features, static and dynamic, employing static and dynamic analysis tools, respectively. Static features are extracted without executing the sample [2]. Dynamic features are also extracted because malware is often obfuscated, limiting static analysis [3]; dynamic features are extracted through sample execution in a virtual environment (sandbox) [4]. All features, static and dynamic, are logged in a feature database. Sisyfos' classifying engine uses the logged features of a sample to classify it as malware or benign, employing several classifiers including ML algorithms and neural networks. Sisyfos' classifying engine approach is analogous to all modern classifiers that employ ML algorithms, e.g. gradient boosting, random forests and support vector machines [5, 6], and, increasingly, DNNs [7, 8].

Effective employment of ML methods is limited by the well-known problem of data availability for training and developing effective models.

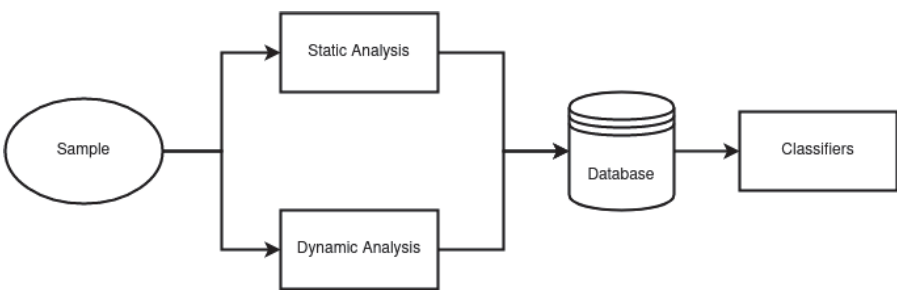

Figure 4.1 Sisyfos architecture: a malware analysis and detection system.

In malware classification, effective and efficient detection models require significant amounts of training data. Although such amounts of data could be collected and made available through data sharing among organizations, there are intellectual property and privacy concerns and constraints that forbid or limit such exchanges. Federated Learning (FL) is a promising method for building effective and efficient detection models in a distributed fashion, using data of different organizations, because it does not require the exchange of source data [9].

FL is employed in two main configurations, cross-device and cross-silo. In cross-device configurations a large number of members (clients) with limited data samples each are coordinating in developing a model. Cross-silo configurations have significantly less members (clients), each with a large population of data samples. FL has been employed for malware detection in cross-device environments, focusing on IoT and Android devices [18, 19, 20]. However, FL in cross-silo configurations has not been explored. Our work focuses on cross-silo FL, considering the requirements of applications and services such as Edge or near-Edge devices and their coordination and collaboration in hierarchies that are being developed internationally.

In this paper, we present cross-silo FL-based malware detection, where the detection model is constructed exploiting horizontal FL and employing a NN. Considering an analysis approach analogous to the one in Sisyfos, we measure the performance in malware detection and evaluate its dependence on several parameters, such as number of clients, repetitions of aggregation steps, dataset size and the percentage of common training data. Our results demonstrate that FL enables high accuracy in malware detection for all members of the federation, irrespective of the size of their own training dataset. This demonstrates an important advantage of FL in malware detection: members of the federation with small training datasets would never achieve independently the high accuracy which they achieve through their participation in the federation.

The paper is organized as follows. Section 1.2 presents an overview of FL and the current state-of-the-art in its employment in malware detection. Section 1.3 presents our cross-silo FL system architecture. Section 1.4 presents our evaluation results and demonstrates the effectiveness of our approach.

4.2 Federated Learning and Related Work

Federated Learning is an emerging machine learning approach that enables the training of AI models in a decentralized manner. Participating clients

collaborate to train ML algorithms under the coordination of a central server, without sharing their private datasets with other parties [10].

To train a federated model a central server distributes to the participating clients an initial model and the training parameters. Then the following steps take place:

1. each participant trains the received model using their private dataset, producing a local model and then sends it to the server;
2. the server aggregates all local models into a global one;
3. the global model is distributed to all clients.

Figure 4.2 illustrates this process which can be repeated for multiple learning steps and stopped when a designated criterion is met.

FL is considered in two different configurations, in general [10]:

- Cross-device: clients are computing systems with limited computing capabilities, varying device availability and small datasets, e.g. IoT devices or smartphones.
- Cross-silo: clients are computing systems with high computational power, high reliability and large datasets (data silos), such as centralized and enterprise systems (typically 2-100).

In traditional centralized machine learning environments, a device or an organization must train a model on its own self-collected data. In practice, these devices or organizations may not have access to sufficiently large data sets and the computing capabilities necessary to train an effective model. Additionally, data privacy concerns and intellectual property rights limit collaboration between parties. FL addresses these challenges by enabling collaboration between multiple parties to jointly train effective ML models with large, diverse datasets collected from all members of the federation [11]. As the produced models are the only information shared among federation members, local data never leave the participating devices enabling data owners to keep their data private. Importantly, FL scales well because additional members can contribute to model training without any burden to other members and with reduced data traffic among them.

FL is employed in several operations of cybersecurity such as attack detection, anomaly detection, trust management, authentication and other IoT related tasks [12, 13, 14, 15]. FL is also effective in malicious URL and Denial-of-Service (DoS) attacks detection [16, 17].

In malware detection, research in FL employment is mainly focused on cross-device FL where federation members (clients) are smartphones [18, 19] or IoT devices [20], while limited effort has been spent on malware detection

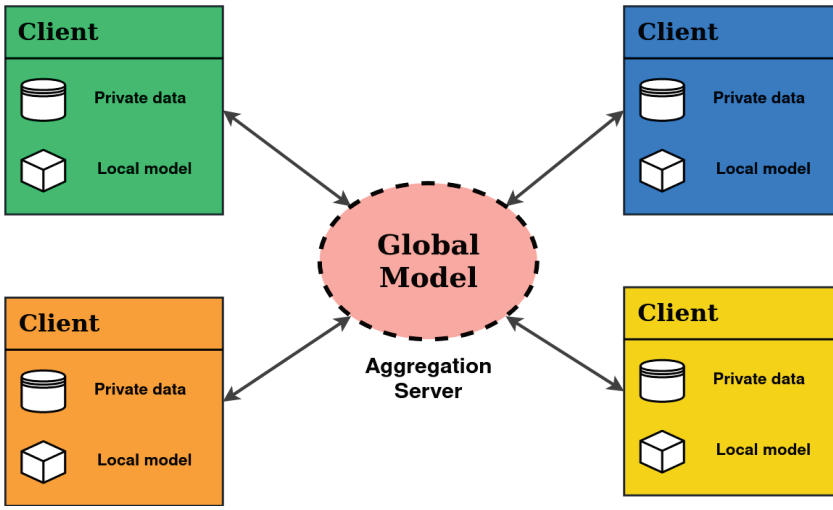

Figure 4.2 Federated Learning configuration.

using cross-silo FL configurations. In our work, we propose a cross-silo FL-based malware detection method, where federation members are different Edge or near-Edge devices deployed to provide security to different organizational networks. The devices collect large amounts of data and have higher computational capabilities relatively to the devices considered in cross-device configurations.

State-of-the-art malware detection approaches employ neural networks architectures to train models for sample classification as either malicious or benign [8, 9]. In our system, we employ a similar neural network [21] that can effectively learn to detect malware from the training data, while being able to fit in Edge or near-Edge devices. Some preliminary results of this work were presented in [26].

4.3 Architecture

Our proposed architecture consists of a FL cross-silo configuration as shown in Figure 4.3. Multiple participating members, indicated as clients, collaboratively train a global malware detection model, and a server is responsible for all communications as well as the aggregation of the global model. Each client owns and trains with some large amount of local private data which it does not share with the other clients nor the server.

In training, each client uses the same feed forward neural network, i.e. the same architecture and training parameters, an increasingly popular method for malware detection [8, 9]. Specifically, we employ a neural network deployed in [21]. We adopt its architecture because it is versatile, widely adopted and can be fitted in devices with limited computing power such as near-edge or edge devices. The model consists of 3 linear layers and a dropout layer. The output layer performs binary classification using a SoftMax layer, classifying a sample as either malicious or benign. We adapt the model in [21] to accommodate our different dataset: EMBER v2 [22] instead of EMBER v1. EMBER v2 is an update on the original EMBER dataset and contains 2381 input features instead of the 2351 used in [21]. The dataset is discussed in more detail in Section 1.4.1.

Our detection system, operates in two modes: (a) training, where multiple clients are using FL to collaboratively train the detection model and (b) detection, where each participating client uses the produced global model to detect malware.

In training mode, the FL-based training process takes place in multiple steps. In each step the following process occurs: (i) each client trains a local model using its own private data, (ii) each client sends the produced local model to the server, (iii) the server aggregates all local models, producing a global model and (iv) the server distributes the global model to all clients. Then, the clients can use the global model to measure the model's performance against their private datasets. The process can be repeated for multiple steps to improve the model's performance further, until a satisfactory model is achieved, considering the time and processing constraints of the clients or until no further accuracy improvement is achieved.

After training is complete, in detection mode, all participating clients have received a copy of the final global model from the server. Each client can use this model to detect malware in their own systems and networks, independently from all other clients. Finally, the clients can return to training mode to refresh and retrain their model with new data.

4.4 Experiments

We evaluate the performance of cross-silo FL measuring the malware detection accuracy on a benchmark dataset containing features from malware and benign files. To further explore the effectiveness of FL, we consider multiple FL training setups measuring how the detection rate is affected by the number of learning steps, the number of participating clients and the commonality in

the participating clients' datasets. Furthermore, to demonstrate the benefits of FL for the participants, we also train a centralized model of the same architecture and parameters and evaluate its performance different training dataset sizes. To conduct the experiments, we employ flower [23], a popular FL framework, for training the federated models, in conjunction with Pytorch [24].

4.4.1 Dataset

In all our experiments we use the EMBER v2 dataset [22], a publicly available benchmark dataset, which contains features extracted from both malware and benign samples using static analysis. We use EMBER because there are no widely available standard datasets; this is a well-known problem in cybersecurity research. Although it does not distribute the sample binary files, due to privacy concerns, EMBER is common choice in malware detection and analysis because of three factors: (i) its sufficient size, (ii) its set of features and (iii) it contains features of malware and benign samples.

EMBER v2 contains 2381 features per sample, extracted using static analysis from 1.1 million Windows Portable Executables (PE). More specifically for training, the dataset contains 600.000 samples labelled as either benign or malicious (300.000 benign and 300.000 malicious), and 300.000 unlabelled samples. The dataset also contains 200.000 samples labelled as either benign or malicious (100.000 benign and 100.000 malicious) to be used as a dedicated benchmark testing set. In our experiments we only use labelled samples, 600.000 for training the neural networks and 200.000 for testing the produced models.

4.4.2 Evaluation results

In the first experiment we consider a FL setup where 2 participating clients train a common model for 10 learning steps using the entire dataset. Thus, each client trains each local model with 300.000 data samples. We measure the accuracy, precision. recall and f1 score of the model on the test set for each step. Table 4.1 summarizes the results of the experiment. For comparison we also train a centralized model on the full dataset, 600.000 data samples and we measure an accuracy of 0,9338 on the test set.

Figure 4.3 plots the accuracy of the FL model for multiple learning steps. The orange line denotes the accuracy of the centralized model as reference trained with the entire EMBER dataset of 600K samples. The results show that the accuracy of FL increases with the increasing number of training loops

Table 4.1 Accuracy on test set of FL model with 2 clients for multiple learning steps.

Number of steps	Accuracy	Precision	Recall	F1 Score
1	0,8711	0,8419	0,9137	0,8763
2	0,9111	0,9012	0,9236	0,9123
3	0,9176	0,9066	0,9312	0,9187
4	0,9194	0,9091	0,9321	0,9205
5	0,9211	0,911	0,9335	0,9221
6	0,9232	0,9123	0,9365	0,9242
7	0,9242	0,9143	0,9361	0,9251
8	0,9261	0,9183	0,9355	0,9268
9	0,9247	0,9143	0,9373	0,9257
10	0,9251	0,9157	0,9365	0,926

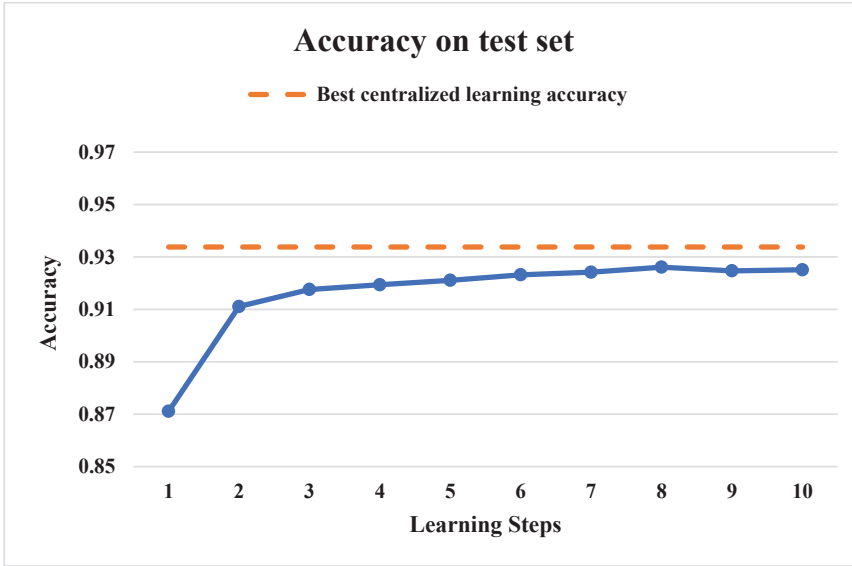

Figure 4.3 Federated Learning model performance for variable training loops.

and importantly reaches the performance of the centralized model and is on par with the results presented in [21]. Additionally, we observe that we make most of the accuracy gains in the first 2 FL training steps for this dataset. Thus, in subsequent experiments we train all FL models for 2 training steps.

Next, we evaluate whether the number of participants influences the accuracy of the produced model. We use the entirety of the EMBER dataset (600.000 samples) and keep the same total dataset size for all experiments, distributing it equally among the participating clients in every case (i.e. for 2

participating clients, each clients holds 300.000 samples and for 5 participating clients, each clients holds 120.000 samples). We run experiments for 2,5,10,15 and 20 participants and measure the accuracy of the produced models on the test set. Table 4.2 summarizes the results of the experiments for 2 learning steps.

Table 4.2 Accuracy on test set of FL model for different number of clients for 2 learning steps.

Number of clients	Accuracy	Precision	Recall	F1 Score
2	0,9103	0,9015	0,9212	0,9112
5	0,9201	0,9156	0,9255	0,9205
10	0,9144	0,9091	0,921	0,915
15	0,9139	0,9089	0,92	0,9144
20	0,9175	0,913	0,9221	0,9175

Figure 4.4 plots the accuracy of the FL model for different number of clients. The orange line denotes the accuracy of the centralized model as reference trained with the entire EMBER dataset of 600K samples. We observe accuracy is effectively independent of the number of clients, suggesting that the malware detection system can scale to more and more participants without accuracy losses.

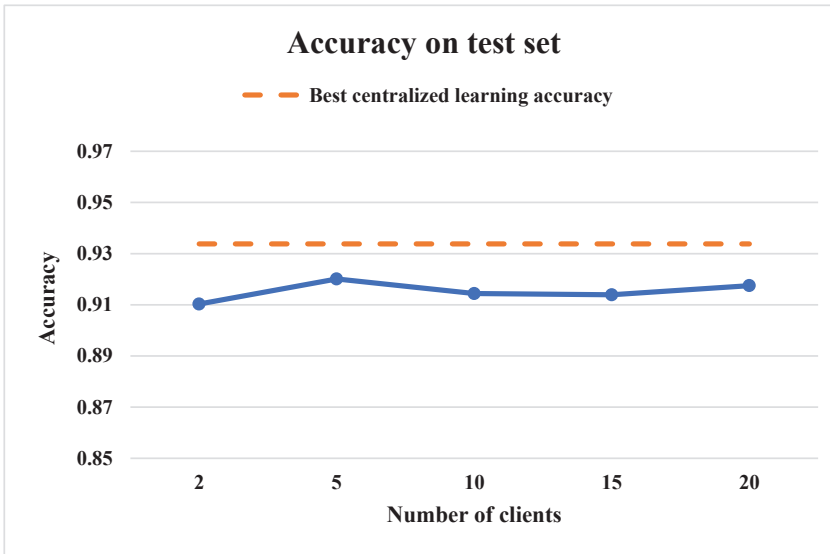

Figure 4.4 Federated Learning model accuracy for different number of clients.

Next, we consider a case where different organizations or different devices in the same organization that participate in a FL setup, have common data samples in their private data. As attackers and malware authors use the same malware samples to infect multiple targets and as organizations process a large amount of malware on daily basis it seems a likely scenario that participants will have some degree of commonality in their private datasets. Thus, we evaluate whether the presence of overlapping samples in the participants' training sets influences the accuracy of the produced FL model. We consider different dataset overlap percentages between the participants for 2 and 10 participating clients. Tables 4.3 and 4.4 present the results of the experiments for 2 and 10 participants respectively for 2 FL training steps.

Table 4.3 Accuracy on test set of FL models for different dataset overlaps for 2 clients.

	2 Clients			
Overlap percentage	**Accuracy**	**Precision**	**Recall**	**F1 Score**
0	0,9111	0,9012	0,9236	0,9123
5	0,9166	0,9139	0,92	0,9169
10	0,9045	0,9039	0,9054	0,9046
15	0,9129	0,9057	0,9218	0,9137
20	0,9114	0,902	0,9231	0,9124
25	0,918	0,9117	0,9256	0,9186
30	0,9125	0,9063	0,9201	0,9131
35	0,9124	0,9098	0,9157	0,9128
40	0,9173	0,9127	0,9228	0,9178
45	0,9319	0,9267	0,9378	0,9322
50	0,9262	0,9197	0,934	0,9268

Table 4.4 Accuracy on test set of FL models for different dataset overlaps for 10 clients.

	10 Clients			
Overlap percentage	**Accuracy**	**Precision**	**Recall**	**F1 Score**
0	0,9144	0,9091	0,921	0,915
5	0,9162	0,9108	0,9229	0,9168
10	0,9154	0,9086	0,9238	0,9161
15	0,9167	0,9096	0,9255	0,9175
20	0,9171	0,9121	0,9233	0,9176
25	0,9169	0,909	0,9267	0,9177
30	0,9171	0,9114	0,9242	0,9177
35	0,9178	0,9138	0,9227	0,9182
40	0,9146	0,9059	0,9254	0,9155
45	0,9179	0,9093	0,9285	0,9188
50	0,9174	0,9114	0,9247	0,918

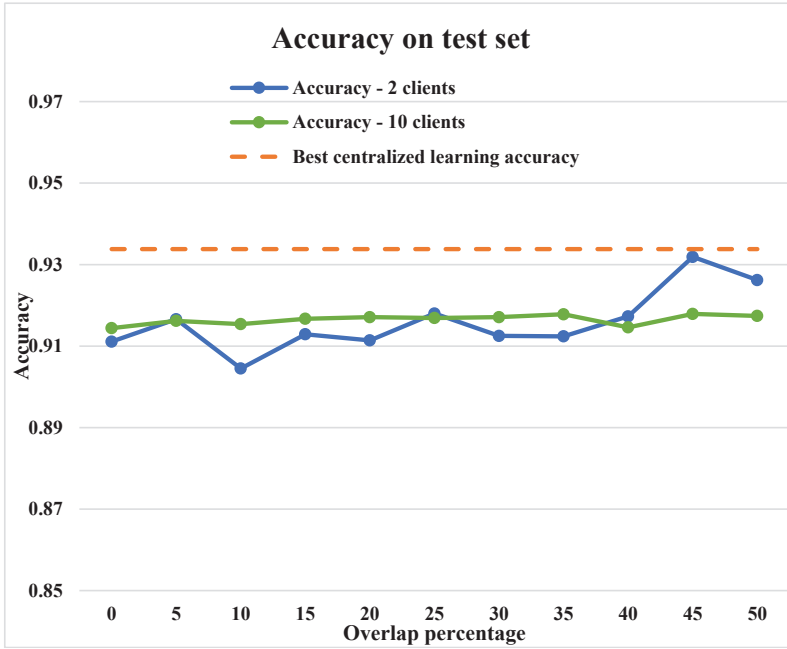

Figure 4.5 Federated model accuracy for different dataset overlaps.

Figure 4.5 plots the accuracy on the test set of the FL models produced as a function of the overlap (common subset) of the clients' training data, i.e. x=5 indicates 5% common data in the client datasets. The blue and green lines depict the accuracy of the models trained by 2 and 10 clients respectively. The orange line denotes the accuracy of the centralized model as reference trained with the entire EMBER dataset of 600K samples. As we observe in both setups, accuracy seems to be effectively independent of the common samples present in the participants' private data even in the extreme case of a 50% overlap, meaning that the produced models do not overfit on the common data.

Finally, to showcase the benefits of FL for organizations, we consider a scenario where a single organization or device is not participating in a FL setup but instead trains its own centralized model using its own private data. We consider organizations of different sizes that have different training data availability. We train a centralized model (no federation present) of the same architecture and parameters as in the previous FL setups with different dataset sizes. Table 4.5 summarizes the results.

Table 4.5 Accuracy on test set of centralized models for different dataset sizes.

Number of samples	Accuracy	Precision	Recall	F1 Score
5000	0,8443	0,8299	0,8662	0,8477
10000	0,8482	0,8428	0,8828	0,8623
50000	0,8949	0,8791	0,9156	0,897
100000	0,9126	0,904	0,9232	0,9135
600000	0,9338	0,9271	0,9417	0,9343

Figure 4.6 plots the accuracy for the centralized (non-federated) system as a function of the dataset size. The blue line denotes the best FL accuracy we measured in our experiments for reference. The results indicate that a data set size of 600K is necessary in the centralized (non-federated) case for achieving high accuracy that reaches above 93% and matches the accuracy of the FL system. This result is the reference accuracy towards which we evaluate the performance of the federated system cases. Importantly, when considering Figure 4.3 as well, the plot demonstrates the benefit of FL for small organizations and near-edge devices with limited data availability that do not have access to large datasets to train centralized models. We also note that even that even organizations and devices that

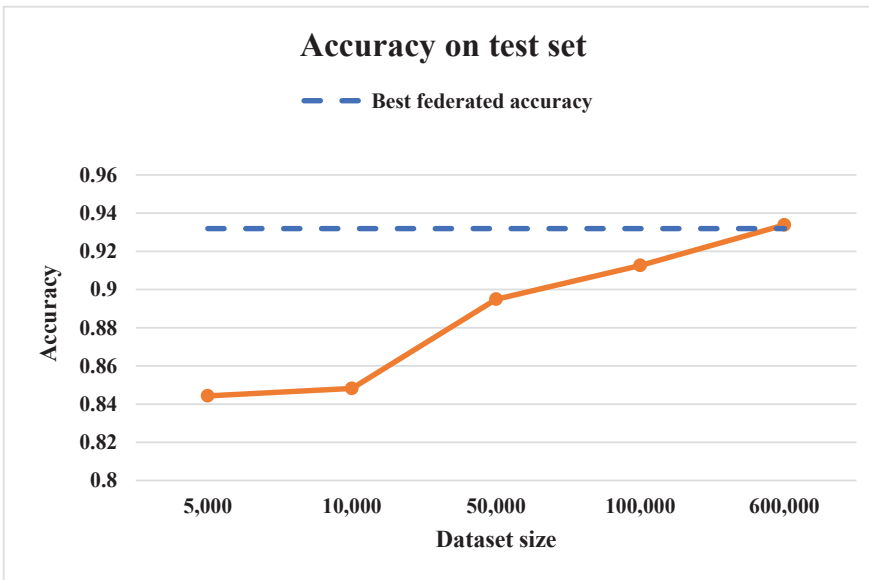

Figure 4.6 Centralized learning model's performance for different dataset sizes.

have access to large datasets can benefit from FL, as a model generated with contributions from multiple participants is trained on a potentially more diverse dataset with malware and benign samples coming from different networks.

4.5 Conclusions

Federated learning constitutes an effective and efficient machine learning technology of classification in malware detection. It provides significant advantages over centralized machine learning solutions, because it enables distributed building of effective malware detection models without source data exchange among members of a federation. We demonstrate that with the adoption of NNs for model training, members of a federation achieve high malware detection accuracy, exceeding, in cases, the accuracy achieved with centralized machine learning methods. Importantly, all members of the federation achieve this accuracy, which would be unattainable if they were limited to training using only their own local data. This advantage also provides a motivation for organizations to participate in federations. In addition to the performance and privacy advantages, federated learning scales well to large federations, due to the low data exchange, and accommodates systems and devices with limited processing power, because the employed NNs of the clients can be efficiently executed even in low-power computational environments. In our work, we considered a centralized aggregating server and Edge or near-Edge devices that provide security in their respective local networks as the federation participants. When considering real word deployments of such setups, additional design parameters should be taken into account. Firstly, the Edge devices should have the computational power and memory to fit and train the chosen NN. In our implementation we specifically used a small model that solves the malware detection task adequately, while requiring low computational power and small memory footprint. Additionally, as discussed in Section 1.3, during training all devices should be available for training and a reliable network connection should be present between each participant and the server. After the training is complete, during inference, no such limitations are present. Finally, the power consumption, latency costs and network communications overhead should be explored; we leave that as future work.

Acknowledgements

Part of D. Serpanos' research was conducted at the Industrial Systems Institute/ATHENA under funding and support by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "2nd Call for H.F.R.I. Research Projects to support Faculty Members & Researchers" (Project Number: 4012).

References

[1] D. Serpanos, P. Michalopoulos, G. Xenos, and V. Ieronymakis, "Sisyfos: A Modular and Extendable Open Malware Analysis Platform." Applied Sciences, 11(7), p. 2980, 3/2021, https://doi.org/10.3390/app11072980.

[2] A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, "Machine Learning Aided Static Malware Analysis: A Survey and Tutorial." In A. Dehghantanha, M. Conti, and T. Dargahi (Eds) Eds., Cyber Threat Intelligence. Advances in Information Security, vol. 70. Cham: Springer International Publishing, 2018, pp. 7–45.

[3] A. Moser, C. Kruegel, and E. Kirda, "Limits of Static Analysis for Malware Detection." In Proceedings of 23rd Annual Computer Security Applications Conference (ACSAC 2007), Miami Beach, FL, USA, 10–14 December 2007, pp. 421–430.

[4] A. Mohaisen, O. Alrawi, and M. Mohaisen, "AMAL: High-fidelity, behavior-based automated malware analysis and classification." Computers & Security, vol. 52, pp. 251–266, Jul. 2015.

[5] W. Li, J. Ge, and G. Dai, "Detecting Malware for Android Platform: An SVM-Based Approach." In Proceedings of 2nd International Conference on Cyber Security and Cloud Computing, New York, NY, USA: IEEE, Nov. 2015, pp. 464–469.

[6] F.C.C. Garcia and F.P. Muga II, "Random Forest for Malware Classification." arXiv, Sep. 25, 2016. [Online]. Available: http://arxiv.org/abs/1609.07770.

[7] R. Vinayakumar, et al. "Robust Intelligent Malware Detection Using Deep Learning." IEEE Access, vol. 7, pp. 46717–46738, 2019.

[8] E. Raff, et al, "Classifying Sequences of Extreme Length with Constant Memory Applied to Malware Detection." AAAI, 35(11), pp. 9386–9394, May 2021, https://doi.org/10.1609/aaai.v35i11.17131.

[9] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning." In Computers & Industrial Engineering, vol. 149, p. 106854, Nov. 2020, https://doi.org/10.1016/j.cie.2020.106854.

[10] P. Kairouz et al., "Advances and Open Problems in Federated Learning," MAL, vol. 14, no. 1–2, pp. 1–210, Jun. 2021, https://doi.org/10.1561/2200000083.

[11] T. Zhang, et al, "Federated Learning for the Internet of Things: Applications, Challenges, and Opportunities." In IEEE Internet of Things, 5(1), pp. 24–29, Mar. 2022, https://doi.org/10.1109/IOTM.004.2100182.

[12] M. Alazab, et al., "Federated Learning for Cybersecurity: Concepts, Challenges, and Future Directions," IEEE Trans. Ind. Inf., 18(5), pp. 3501–3509, May 2022, https://doi.org/10.1109/TII.2021.3119038.

[13] M. Venkatasubramanian, A. H. Lashkari, and S. Hakak, "IoT Malware Analysis Using Federated Learning: A Comprehensive Survey." In IEEE Access, vol. 11, pp. 5004–5018, 2023.

[14] T. D. Nguyen, et al, "DÏoT: A Federated Self-learning Anomaly Detection System for IoT." In Proceedings IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA: IEEE, Jul. 2019, pp. 756–767. https://doi.org/10.1109/ICDCS.2019.00080.

[15] V. Mothukuri, et al, "Federated-Learning-Based Anomaly Detection for IoT Security Attacks." IEEE Internet of Things, 9(4), pp. 2545–2554, Feb. 2022, https://doi.org/10.1109/JIOT.2021.3077803.

[16] E. Khramtsova, C. Hammerschmidt, S. Lagraa, and R. State, "Federated Learning For Cyber Security: SOC Collaboration For Malicious URL Detection." In Proceedings IEEE 40th International Conference on Distributed Computing Systems (ICDCS), Singapore, Nov. 2020, pp. 1316–1321. https://doi.org/10.1109/ICDCS47774.2020.00171.

[17] R. Doriguzzi-Corin and D. Siracusa, "FLAD: Adaptive Federated Learning for DDoS Attack Detection." arXiv, Aug. 23, 2022. [Online]. Available: http://arxiv.org/abs/2205.06661.

[18] R. Gálvez, V. Moonsamy, and C. Diaz, "Less is More: A privacy-respecting Android malware classifier using federated learning." In Proceedings on Privacy Enhancing Technologies, 2021(4), pp. 96–116, Oct. 2021, https://doi.org/10.2478/popets-2021-0062

[19] R.-H. Hsu et al., "A Privacy-Preserving Federated Learning System for Android Malware Detection Based on Edge Computing." In Proceedings 15th Asia Joint Conference on Information Security (AsiaJCIS), Taipei, Taiwan, Aug. 2020, pp. 128–136.

[20] V. Rey, P. M. S. Sánchez, A. H. Celdrán, G. Bovet, and M. Jaggi, "Federated Learning for Malware Detection in IoT Devices," Computer Networks, vol. 204, p. 108693, Feb. 2022, https://doi.org/10.1016/j.comnet.2021.108693.

[21] S. Pramanik and H. Teja, "EMBER - Analysis of Malware Dataset Using Convolutional Neural Networks." In Proceedings 2019 Third International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, Jan. 2019, pp. 286–291, https://doi.org/10.1109/ICISC44355.2019.9036424.

[22] H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models." arXiv, Apr. 16, 2018, https://doi.org/10.48550/arXiv.1804.04637.

[23] D. J. Beutel et al., "Flower: A Friendly Federated Learning Research Framework." arXiv, Mar. 5, 2022, https://doi.org/10.48550/arXiv.2007.14390.

[24] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library." arXiv, Dec. 03, 2019, https://doi.org/10.48550/arXiv.1912.01703.

[25] "VirusTotal." https://www.virustotal.com/gui/.

[26] D. Serpanos and G. Xenos, "Federated Learning in Malware Detection," 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), Sinaia, Romania, 2023, pp. 1-4, https://doi.org/10.1109/ETFA54631.2023.10275578.