Explainable Malware Detection by means of Federated Machine Learning

Giovanni Ciaramella*†, Fabio Martinelli‡, Antonella Santone§, Francesco Mercaldo§†
* IMT School for Advanced Studies Lucca, Lucca, Italy
Email: giovanni.ciaramella@imtlucca.it
† Institute for Informatics and Telematics, National Research Council of Italy (CNR), Pisa, Italy
Email: {giovanni.ciaramella, francesco.mercaldo}@iit.cnr.it
‡ Institute for High Performance Computing and Networking, National Research Council of Italy (CNR), Rende, Italy
Email: fabio.martinelli@icar.cnr.it
§ University of Molise, Campobasso, Italy
Email: {francesco.mercaldo, antonella.santone}@unimol.it

2025 IEEE International Conference on Cyber Security and Resilience (CSR) | 979-8-3315-3591-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/CSR64739.2025.11130004

***Abstract*—Due to the escalating proliferation of malware in the Microsoft Windows environment, effective detection methods have become crucial. Moreover, many existing approaches lack transparency and do not adequately manage personal data in compliance with government regulations. This paper proposes a method to detect malware in the Windows domain by leveraging Federated Machine Learning and explainability. Specifically, we transformed a dataset of malicious and trustworthy Portable Executable and Object Linking and Embedding files belonging to the Windows environment into grayscale images. As the next step, we train multiple models on non-Independent and Identically Distributed data to better represent a real-world scenario, both with and without Differential Privacy norm, to evaluate its impact on privacy and performance. After selecting the most accurate models, we employed the Gradient-weighted Class Activation Mapping algorithm to visualize the most influential features, enhancing interpretability and trust in predictions.**

***Index Terms*—windows, malware, artificial intelligence, federated machine learning, explainability, security**

I. Introduction

Currently, Microsoft Windows is the most widely used desktop operating system (OS) in the world. As shown in a report published by Statcounter in February 2025[1], the OS developed by Microsoft is used by up to 70.62% of global users, and it is the primary platform for personal, business, and enterprise tasks. Due to its massive adoption, Windows has become one of cybercriminals' favorite targets. In a report published by Statista, in 2023, 92% of attacks perpetrated against Windows users were performed using ransomware[2]. Moreover, during 2024 Kaspersky detected 467,000 malicious files daily[3]. The primary motivation behind these attacks is financial gain for cybercriminals. Ransomware provides a direct way to extort money from victims, but it is not the only method. Personal data has become a valuable commodity, often sold on the dark web for profit. To address these growing threats, several regulations, such as the General Data Protection Regulation (GDPR)[4] in the European Union and HIPPA in the United States of America[5] have been introduced over the years to enhance data security and privacy protections. Over the years, several methods have been proposed to avoid cyberattacks, using different techniques like Deep Learning [1]–[3] and Machine Learning [4]–[6]. In 2017, the Federated Machine Learning (FML) approach was introduced to comply with these regulations [7]. In a nutshell, FML avoids transferring private information from the client to the central server. However, the information stays locally, while only the model weights are shared from clients to a central server, minimizing the risk of data exposure and preserving client data privacy. Over the years, different methodologies have also been applied to the FML scenario to increase security, such as Differential Privacy (DP), which aims to add noise to the original dataset or the learning parameters. Thanks to the latter, possible attackers could not extract sensitive information in the training set [8], [9]. Moreover, one of the most important problems related to all artificial intelligence branches is interpreting the outcomes obtained. For this reason, researchers have proposed several explainability tools over the years, such as Local Interpretable Model-Agnostic Explanations [10] for Machine Learning models and Gradient-weighted Class Activation Mapping (Grad-CAM) [11] for Deep Learning.

Starting from these considerations, we present a malware detection method related to the Microsoft Windows environment using FML in this paper. Specifically, we converted 20,000 Portable Executable (PE) and Object Linking and Embedding (OLE) malware and trusted files into images to create the dataset. The latter was used for extensive Convolutional Neural Networks (CNNs) experiments. Additionally, we trained several models with and without Differential Privacy to enhance security. After completing the experimental phase and identifying the best models, we utilized the Grad-CAM algorithm to generate heatmaps, highlighting the regions that contributed the most to specific predictions. Our main contri-

[1]https://gs.statcounter.com/os-market-share/desktop/worldwide
[2]https://www.statista.com/statistics/1498850/most-targeted-operating-systems-with-ransomware/
[3]https://www.kaspersky.com/about/press-releases/the-cyber-surge-kaspersky-detected-467000-malicious-files-daily-in-2024
[4]https://gdpr.eu/
[5]https://www.hhs.gov/hipaa

butions are as follows:

- We propose a malware detection method for the Windows environment using Federated Machine Learning using a dataset of images retrieved from 20,000 malicious and trustworthy Portable Executable and Object Linking and Embedding files.
- We train models with and without Differential Privacy, evaluating the trade-off between performance and security.
- We incorporate explainability into the malware detection process by using Grad-CAM to generate heatmaps that highlight the most influential regions for model predictions.

To the best of our knowledge, this is the first study to incorporate explainability into image-based malware detection by utilizing images extracted from PE and OLE files to train models using a Federated Machine Learning approach.

The paper is organized as follows: the following section presents a state-of-the-art overview of Federated Machine Learning in Android malware detection. Section III details the proposed method, while Section IV showcases the experimental evaluation results obtained from real-world Microsoft Windows files. Finally, the conclusion and potential directions for future research are discussed in the last section.

II. Related Work

This section provides an overview of the literature on malware detection, focusing on approaches with and without Federated Machine Learning.

Lin *et al.* in [12] proposed a method based on FML to identify malware in Windows scenarios using a dataset of almost 20,000 samples. Concluded the dataset composition phase, they leveraged SVM and LSTM approaches to train several models, achieving high-accuracy results. Unlike them, in our proposed approaches, we converted PE and OLE files into images using a Python script written by the authors and trained several models using Differential Privacy and not adopting a custom version of MobileNetV3. Moreover, we applied the Grad-CAM algorithm for explainability purposes on models that reached the best accuracy.

Researchers in [13] presented a ransomware detector to identify ransomware in the Windows environment. In detail, once collected the dataset of less than 4,000 samples, they employed Support Vector Machines and Random Forest to perform classification tasks, achieving interesting results in terms of accuracy. Differently, in our proposed paper, we created a bigger dataset (20,000) by converting samples into images. Moreover, we compared the Differential Privacy norm to ensure privacy, generating a visual explanation of models with this norm and not using the Grad-CAM algorithm.

Ullah *et al.* in [14] proposed a malware detection approach using Federated Machine Learning. The authors employed a deep CNN written by them to train their models using images retrieved from malware and trusted applications. Moreover, they employ data augmentation to balance the number of samples in specific classes. Similarly to our approach, they used non-IID data to represent a real-world scenario better. However, we enhanced security by applying Differential Privacy norms and introduced a comparison using Grad-CAM.

Authors in [15] presented a method for a multiclass classification of Android applications converted into images leveraging FML. In detail, they employed Identically Independently Distributed data to distribute samples, achieving interesting results in accuracy using both Batch Normalization and Group Normalization. In our proposed approach, we employed a dataset of images retrieved from files belonging to the Windows environment, which were distributed among clients using a non-identically Independently Distributed approach to better represent a real-world scenario. Moreover, we trained several models using Differential Privacy and not by applying the best Grad-CAM algorithm to explain a certain classification visually.

Çıplak *et al.* in [16] illustrated a Federated Machine Learning detector using a dataset named CIC-MalMem-2022, which includes obfuscated malware and trusted files. In detail, they employed feedforward neural networks and long short-term memory methods, including four non-federated models achieving interesting performances both for multiclass and binary classification. Unlike them, we conducted binary classification using a customized MobileNetV3 architecture, integrating Differential Privacy instead of norm-based techniques to improve privacy in FML. Moreover, concluding the training, validation, and testing phases, we identified the best models and applied the Grad-CAM algorithm to them to provide explainability.

Authors in [17] proposed a method to identify malware, including *.pdf, .doc,* and *.htm* extensions, not only executable files. In detail, they converted files into RGB images using an online tool, and then, after the dataset creation, they employed a Deep Learning architecture to perform classification. At the end of the test phase, they obtained interesting performances in terms of accuracy for files with *.pdf* and *.doc* extensions. Unlike these models, we employed a Federated Machine Learning approach combined with differential privacy to propose a privacy-preserving alternative to traditional deep learning methods. Moreover, we considered only PE and OLE files belonging to the Windows environment.

Images are widely adopted not only for malware detection but also for intrusion detection. Rose *et al.* in [18] proposed a method based on network traffic profiling and machine learning for IoT networks. Using network traffic profiling to monitor device behavior and applying machine learning algorithms to detect anomalies, the authors reached a high accuracy value and a low false positive rate. Different from them, our approach focuses on detecting malware in a Windows environment using a Federated Machine Learning framework. We also incorporate explainability to identify which regions of an image contributed to a given classification.

III. The Method

In this section, we present the proposed method for malware detection in the Windows environment by leveraging Federated

Machine Learning. Figure 1 reports the workflow pipeline followed to achieve outcomes.

A. Image Generation and Dataset Composition

As the first step of the proposed method, we identified a collection of trustworthy and malicious PE and OLE files. Once the data was collected files, we converted them into grayscale images leveraging a Python script developed by the authors. Specifically, the latter operates by first converting each binary file byte into a numerical value ranging from 0 to 255, corresponding to grayscale intensity levels [15], [19]. The final PNG image represents each byte as a single grayscale pixel. This approach ensures a visual encoding of the binary data. In conclusion, to improve the efficiency of the learning process, each final image was resized to a standardized dimension of 300×300 pixels. Figure 2 exhibits two examples of images obtained as results. In detail, Figure 2a displays an image obtained from a malware sample, while Figure 2b shows a trusted sample.

B. Experiments and Data Analysis

Concluded the dataset creation phase, as the following step, we built several models leveraging FML with and without DP. The latter's usage relies on the goal of developing a model that can be effectively deployed in real-world applications while preserving user privacy. By incorporating DP, the model ensures that individual data points remain untraceable, reducing the risk of exposure and unauthorized access. Moreover, to best represent real-world scenarios, we employed a specific type of distribution *i.e.,* non-IID explicitly using the Dirichlet approach proposed by Li *et al.* in [20]. Figure 3 exhibits the distribution of the samples among the 20 clients employed. Moreover, only 17 participated in the learning process during each training round, enhancing the model's learning capacity. This partial participation strategy is commonly employed in the FML scenario to reduce communication overhead and to improve the scalability. On the other hand, Table I showcases the number of samples for each class used in each client during the training and validation process.

TABLE I: Number of samples per client in the dataset.

Client	Malware	Trusted	Client	Malware	Trusted
1	178	590	*11*	499	269
2	409	359	*12*	258	510
3	627	141	*13*	659	109
4	768	0	*14*	609	159
5	520	248	*15*	609	159
6	106	662	*16*	522	246
7	768	0	*17*	176	592
8	681	87	*18*	103	665
9	741	27	*19*	1	767
10	497	271	*20*	1	767

Regarding the CNN used to conduct experiments, we utilized a customized version of MobileNetV3, adding Dense and Dropout layers to improve the regularization and reduce overfitting. Moreover, we utilized the ImageNet[6] weights. Next, we employed a grid search algorithm to optimize hyperparameters, exploring learning rates of 1E-04 and 5E-04, 20 and 30 rounds, and 5 and 10 client epochs per round. To ensure a fair comparison, we maintained a fixed image size of $224 \times 224 \times 3$, noise multiplier of 0.006, norm l2 16.0, standard deviation of 0.2, and a distribution of 20 clients, with 17 participants per round. Table II presents the best three combinations of hyperparameters that yielded the best accuracy results with and without using Differential Privacy.

C. Explainability

Upon concluding the experimental procedure, we implemented the Grad-CAM algorithm introduced by Selvaraju *et al.* [11] on the best models, both with and without DP. The algorithm utilizes gradients to identify the regions of the image that most significantly influenced the model's decision. Specifically, it leverages the gradients flowing into the final convolutional layer, which retains spatial information crucial for interpretation. The Grad-CAM algorithm then uses this information to generate a heatmap that highlights the most relevant areas of the image.

IV. Results

In this section, we present the results obtained using the proposed approach. In detail, the best outcomes obtained after the test phases are discussed in Section IV-A, while Section IV-B displays the results achieved by applying the Grad-CAM algorithm. The experiments reported in this research article have been executed using the following hardware and software characteristics: Intel Xeon Gold 6140M CPU at 2.30GHz, 64GB RAM, and Ubuntu 22.04.01 LTS distro as Operating System.

A. Models Evaluation

In the proposed method, we employed a public dataset of Portable Executable and Object Linking and Embedding files available on a GitHub repository[7]. In detail, that dataset comprises several executable files for the Windows environment divided into classes *i.e.,* malware and trusted. We downloaded the dataset and applied a Python script to convert files into grayscale images to compose the final dataset. Specifically, the dataset was constructed using 10,000 images per class, resulting in 20,000 samples. In the following step, the dataset was split into training, validation, and test subfolders using a common splitting of 80-10-10. As the next step, we began conducting several experiments using the set of hyperparameters listed in Table II. The results of these experiments are presented in Table IV. Both tables present only the top six hyperparameters and the results achieved after the test phase.

Regarding results, Run 1 achieved the best accuracy values with and without DP. In detail, the best model using DP reached an accuracy value of 0.877 after training for 20 rounds, with 5 client epochs per round and a learning rate of 5E-04. Moreover, it achieved a loss value of 0.389, precision and recall values of 0.877, and an AUC of 0.940. The model showed

[6]https://www.image-net.org/

[7]https://github.com/iosifache/DikeDataset

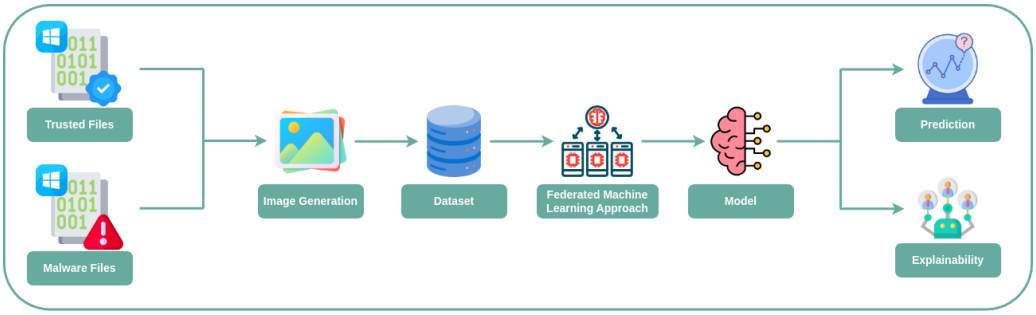
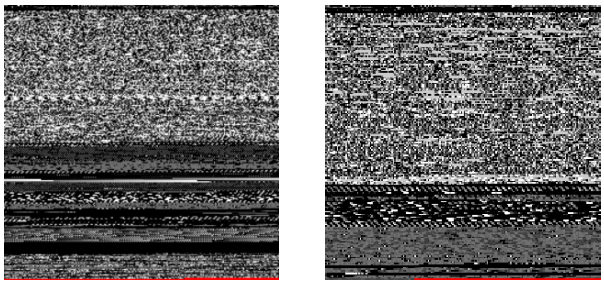


Fig. 1: The proposed Federated Machine Learning-based method for detecting malware and trusted files in Windows PE and OLE.

TABLE II: Hyperparamters adopted to train Federated Machine Learning models.

Run	Norm Type	TC	CR	Distribution	Img Size	TR	CER	Learning Rate	Noise Multiplier	Norm L2	Standard Deviation
1		20	17	non-IID	224	20	5	5E-04	0.006	16.0	0.2
2	Differential	20	17	non-IID	224	20	5	1E-04	0.006	16.0	0.2
3		20	17	non-IID	224	30	10	1E-04	0.006	16.0	0.2
1		20	17	non-IID	224	30	5	1E-04	-	-	-
2	None	20	17	non-IID	224	20	5	1E-04	-	-	-
3		20	17	non-IID	224	30	5	5E-04	-	-	-

Legend: **TC**: Total Client; **CR**: Client per Round; **TR**: Total Rounds; **CER**: Client Epochs per Round;

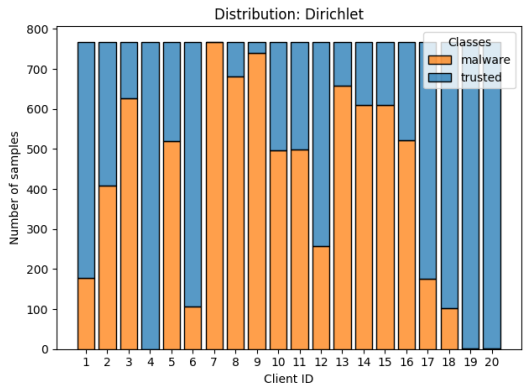


(a) Malware sample generated after file conversion

(b) Trusted sample generated after file conversion

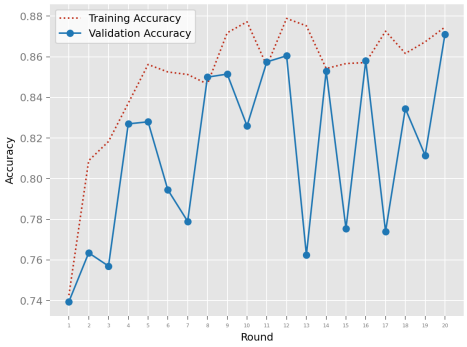
Fig. 2: Malware and Trusted file representation.

Fig. 3: Distribution of samples across clients in the dataset.

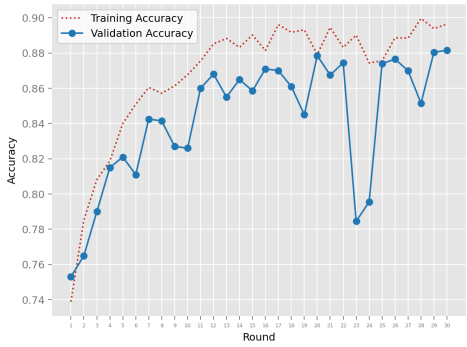


engaging performances for both classes, achieving identical performances in terms of loss (0.386), accuracy (0.876), and AUC (0.876). Regarding precision, the model showed higher performances regarding the malware class, reaching a value of 0.908, while 0.848 regarding the trusted class. On the other hand, the Recall and F-Measure scores showed better performance in the trusted class, achieving 0.916 and 0.881, respectively, compared to 0.837 and 0.871 in the malware class. We concluded the training and validation phases to avoid overfitting and created a plot to show the performances obtained. In detail, Figure 4a illustrates the accuracy values achieved, while Figure 5a presents the corresponding loss values. In the plots, red lines represent the values obtained during the training phase, whereas blue lines indicate the results from the validation phase. Additionally, we plot the difference between accuracy values recorded in both phases. Figure 6a shows the latter, where the maximum gap occurs during the 13th round, slightly higher than 0.10, indicating that the model effectively identifies samples while avoiding overfitting. Furthermore, to better understand the classification capabilities of the model, we also calculated the confusion matrix, which is reported in Figure 7a. In detail, the model correctly identified 837 malware samples, misclassifying 163 as trusted. On the other hand, it accurately classified 916 trusted samples, with 84 misclassified as malware. As it is possible to observe from the metrics reported before, the model achieved a slightly higher recall for trusted samples, indicating it is more effective at recognizing trusted instances than malware. However, the false positive rate remains a concern in security applications, as misclassifying malware as trusted could pose a significant risk.

Concerning the model trained without the DP norm, we observed slightly higher accuracy (0.886) and loss (0.283).

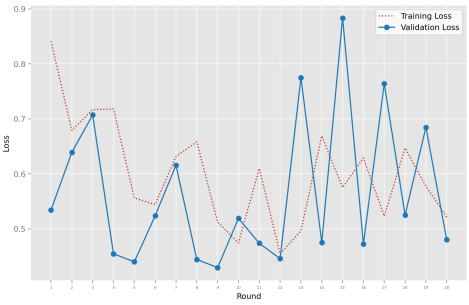


(a) Accuracy achieved in each round during the training and validation phases using Differential Privacy.

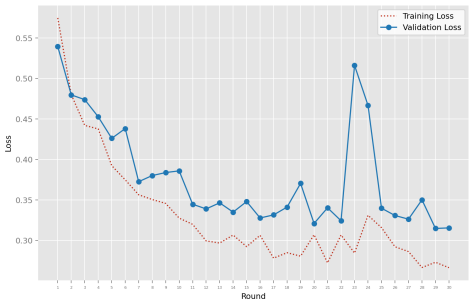


(b) Accuracy achieved in each round during the training and validation phases without using Differential Privacy.

Fig. 4: Accuracy comparison of models with and without Differential Privacy during training and validation.

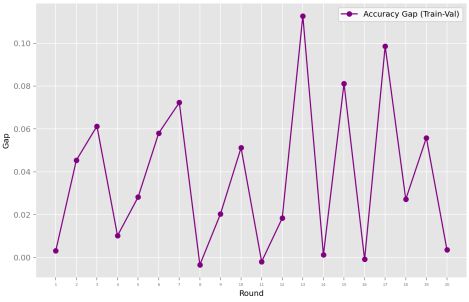


(a) Loss achieved in each round during the training and validation phases using Differential Privacy.

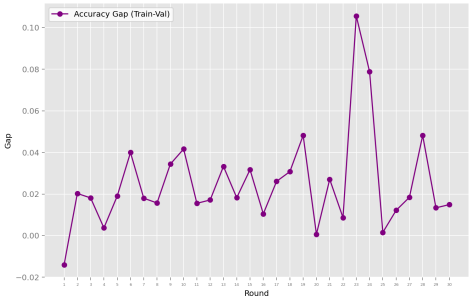


(b) Loss achieved in each round during the training and validation phases without using Differential Privacy.

Fig. 5: Loss comparison of models with and without Differential Privacy during training and validation.



(a) Difference achieved in each round during the training and validation accuracy phase using Differential Privacy.



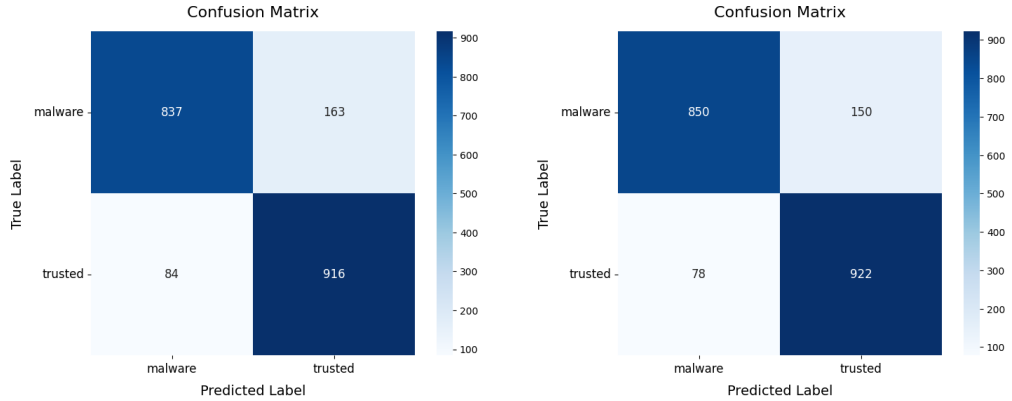
(b) Difference achieved in each round during the training and validation accuracy phase without using Differential Privacy.

Fig. 6: Difference comparison in training and validation accuracy per round for models with and without Differential Privacy.

TABLE III: Results obtained after test set execution.

Run	Norm Type	Loss	Accuracy	Precision	Recall	F-Measure	AUC
1		0.389	0.877	0.877	0.877	0.877	0.94
2	Differential	0.299	0.874	0.874	0.874	0.874	0.946
3		0.332	0.873	0.873	0.873	0.873	0.943
1		0.283	0.886	0.886	0.886	0.886	0.952
2	None	0.313	0.876	0.876	0.876	0.876	0.943
3		0.422	0.859	0.859	0.859	0.859	0.935

Additionally, precision, recall, and AUC generated promising results of 0.886 and 0.952. The model showed identical values for loss, accuracy, and AUC for individual classes, all at 0.282, 0.886, and 0.886, respectively. However, it demonstrated higher precision in malware detection, achieving 0.915 compared to 0.860 for trusted samples. On the other hand, the model performed better regarding recall and F-measure for



(a) Confusion matrix obtained with Differential Privacy.

(b) Confusion matrix obtained without Differential Privacy.

Fig. 7: Confusion matrix obtained after the testing phase with and without Differential Privacy.

trusted samples, with values of 0.922 and 0.889, compared to 0.850 and 0.881 for malware detection. Similar to the model trained using DP, we created plots using metrics obtained during the training and validation phases. In detail, Figure 4b illustrates the accuracy trends across the rounds, while Figure 5b presents the corresponding loss trend. To assess overfitting, we also calculated the difference in accuracy between the training and validation phases for each round, which is shown in Figure 6b. From this plot, we can observe that the model exhibited the most significant difference in the 23rd round, with a gap of 0.10, indicating strong performance during the training phase without significant overfitting. Once we concluded the training and validation phases, we also calculated the confusion matrix, which is reported in Figure 7b. The latter shows that the model correctly identified 922 trusted samples as trusted and 850 malware samples as malware. However, it misclassified 78 trusted samples as malware and 150 malware samples as trusted. Also, the model demonstrated more substantial capabilities in detecting trusted samples than malware in this case. Table IV presents the metrics obtained during testing for each class, comparing the performance of the best model with and without the use of Differential Privacy.

B. Grad-CAM Visual Explainability

Once we concluded the phases related to the training, validation, and testing of several models and identified which provided the best results in terms of accuracy, we applied the Grad-CAM algorithm to them. In detail, we leveraged the explainability of the models we adopted, rather than applying Differential Privacy, to assess whether the models could effectively identify similar areas. After applying Grad-CAM, we generated a heatmap for each submitted sample, using three distinct colors to highlight the most relevant areas for the model's prediction. Yellow, green, and blue serve specific purposes in classification: yellow and green indicate the most important regions, with green specifically emphasizing smaller areas, while blue marks less relevant regions. To enhance visualization, we present the output as a composite image consisting of the original image, the generated heatmap, and the heatmap overlaid on the original image.

Figure 8 showcases the heatmap obtained using the same samples on the left (Figure 8a) using the DP norm and on the right (Figure 8b) without using the latter. As it is possible to denote, both models classified the samples as malware with a higher accuracy level. Unlike the model with Differential Privacy, the second model identified the sample under consideration with slightly lower accuracy (99.29% instead of 100%).

On the other hand, identical areas have been identified as important for classification purposes. An identical situation happened in the trusted sample, as reported in Figure 9. In detail, the first model (trained using Differential Privacy) achieved a higher accuracy of 100% (Figure 9a), unlike the model in which that norm was not used (Figure 9b).

Misclassifications happened in some cases, as reported in the discussion provided in the section above. Figure 10 reports two heatmaps generated using the Grad-CAM algorithm using the two different models in which the first model (trained using DP) wrongly identified a malware sample as trusted (Figure 10a), while even similar areas have been identified, the second model (Figure 10b) correctly classified the samples as trusted albeit with low accuracy (54%).

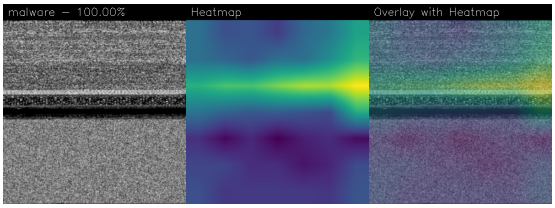
In conclusion, both models misclassified certain samples in some cases. An example is shown in Figure 11, where both models identified the same area but incorrectly classified a trusted sample as malware.

V. Conclusion and Future Work

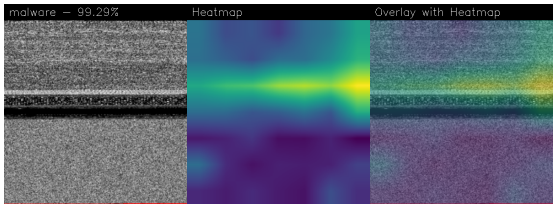
Effective detection methods have become essential with the increasing prevalence of malware in the Windows environment. However, many existing approaches do not explain the decision made by a model, preventing end users from understanding the latter. Additionally, managing personal data in compliance with evolving government regulations remains a significant challenge. To tackle these issues, in this paper, we proposed a novel method for detecting malware in the

TABLE IV: Best model evaluation metrics for each class (Malware and Trusted) across loss, accuracy, precision, recall, F-measure, and AUC.

Norm	Class	Loss	Accuracy	Precision	Recall	F-Measure	AUC
Differential Privacy	Malware	0.386	0.877	0.909	0.837	0.871	0.877
	Trusted	0.386	0.877	0.849	0.916	0.881	0.876
None	Malware	0.283	0.886	0.916	0.850	0.882	0.886
	Trusted	0.283	0.886	0.860	0.922	0.890	0.886

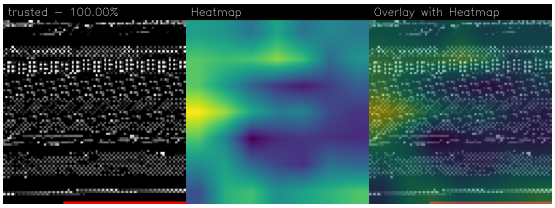


(a) Heatmap generated by the model trained with Differential Privacy, correctly identifying a malware sample.

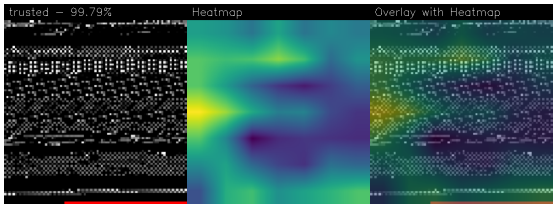


(b) Heatmap generated by the model trained without Differential Privacy, correctly identifying a malware sample.

Fig. 8: Example of heatmaps generated using the Grad-CAM algorithm, where both models identified similar areas and correctly classified the sample as malware.

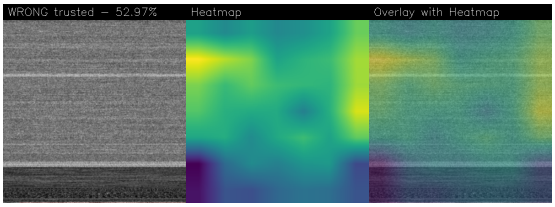


(a) Heatmap generated by the model trained with Differential Privacy, correctly identifying a trusted sample.

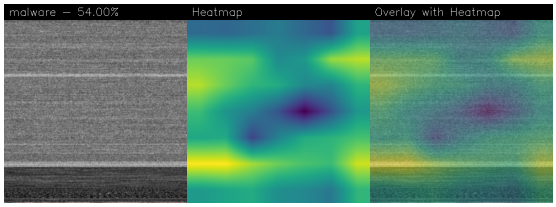


(b) Heatmap generated by the model trained without Differential Privacy, correctly identifying a trusted sample.

Fig. 9: Example of heatmaps generated using the Grad-CAM algorithm, where both models identified similar areas and correctly classified the sample as trusted.



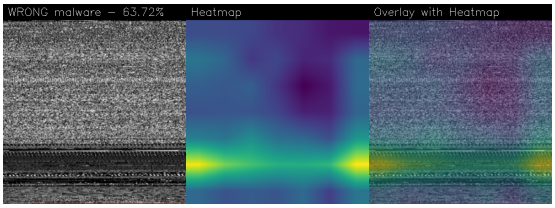
(a) Heatmap generated by the model trained with Differential Privacy, incorrectly classifying a malware sample as trusted.



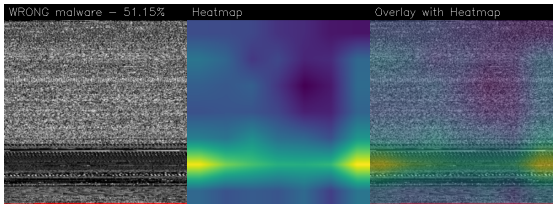
(b) Heatmap generated by the model trained without Differential Privacy, correctly identifying a trusted sample.

Fig. 10: Example of heatmaps generated using the Grad-CAM algorithm, where the model trained with Differential Privacy failed to classify the sample as malware, while the model trained without Differential Privacy correctly identified it.

Windows domain by leveraging Federated Machine Learning and explainability techniques. Specifically, in the proposed method, we trained several models using images generated from PE and OLE files using a custom Python script. After constructing the dataset, we employed non-IID data to simulate real-world scenarios better and trained a custom version of MobileNetV3. To enhance privacy, we also applied the Differential Privacy technique. Our experimental results achieved interesting accuracy results, reaching 0.877 with Differential Privacy and 0.886 without it. Furthermore, we employed the Grad-CAM algorithm to interpret model predictions, providing insights into decision-making. To the best of our knowledge, this is the first study to integrate explainability into image-based malware detection using images extracted from PE and OLE files to train models using the FML approach. As future work, we plan to extend our approach by experimenting with other Convolutional Neural Networks, incorporating deeper evaluations of hyperparameters. The latter approach



(a) Heatmap generated by the model trained with Differential Privacy, incorrectly classifying a trusted sample as malware.



(b) Heatmap generated by the model trained without Differential Privacy, incorrectly classifying a trusted sample as malware.

Fig. 11: Example of heatmaps generated using the Grad-CAM algorithm, where both models identified similar areas and wrongly classified the sample as malware.

will enable us to optimize models for better performance. Moreover, we will consider adopting other state-of-the-art norms to ensure privacy and use different CAM algorithms to improve explainability and robustness.

ACKNOWLEDGMENT

This work has been partially supported by EU DUCA, EU CyberSecPro, SYNAPSE, PTR 22-24 P2.01 (Cybersecurity) and SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the EU - NextGenerationEU projects, by MUR - REASONING: foRmal mEthods for computAtional analySis for diagnOsis and progNosis in imagING - PRIN, e-DAI (Digital ecosystem for integrated analysis of heterogeneous health data related to high-impact diseases: innovative model of care and research), Health Operational Plan, FSC 2014-2020, PRIN-MUR-Ministry of Health, the National Plan for NRRP Complementary Investments D^3 4 Health: Digital Driven Diagnostics, prognostics and therapeutics for sustainable Health care, Progetto MolisCTe, Ministero delle Imprese e del Made in Italy, Italy, CUP: D33B22000060001, FORESEEN: FORmal mEthodS for attack dEtEction in autonomous driviNg systems CUP N.P2022WYAEW and ALOHA: a framework for monitoring the physical and psychological health status of the Worker through Object detection and federated machine learning, Call for Collaborative Research BRiC -2024, INAIL.

REFERENCES

[1] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A review of android malware detection approaches based on machine learning," *IEEE access*, vol. 8, pp. 124 579–124 607, 2020.

[2] G. Ciaramella, G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Explainable ransomware detection with deep learning techniques," *Journal of Computer Virology and Hacking Techniques*, vol. 20, no. 2, pp. 317–330, 2024.

[3] G. Ciaramella, F. Mercaldo, and A. Santone, "Dynamic analysis for explainable fine-grained android malware detection," in *International Workshop on Security and Trust Management*. Springer, 2024, pp. 110–127.

[4] A. Mahindru and A. L. Sangal, "Mldroid—framework for android malware detection using machine learning techniques," *Neural Computing and Applications*, vol. 33, no. 10, pp. 5183–5240, 2021.

[5] J. Senanayake, H. Kalutarage, and M. O. Al-Kadri, "Android mobile malware detection using machine learning: A systematic review," *Electronics*, vol. 10, no. 13, p. 1606, 2021.

[6] P. He, L. Cavallaro, and S. Ji, "Defending against adversarial malware attacks on ml-based android malware detection systems," *arXiv preprint arXiv:2501.13782*, 2025.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[8] A. El Ouadrhiri and A. Abdelhadi, "Differential privacy for deep and federated learning: A survey," *IEEE access*, vol. 10, pp. 22 359–22 380, 2022.

[9] C. Peluso, G. Ciaramella, F. Mercaldo, A. Santone, and F. Martinelli, "A federated learning-based android malware detector through differential privacy," in *International Conference on Computer Aided Systems Theory*. Springer, 2024, pp. 307–319.

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[11] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[12] K.-Y. Lin and W.-R. Huang, "Using federated learning on malware classification," in *2020 22nd international conference on advanced communication technology (ICACT)*. IEEE, 2020, pp. 585–589.

[13] A. Vehabovic, H. Zanddizari, F. Shaikh, N. Ghani, M. S. Pour, E. Bou-Harb, and J. Crichigno, "Federated learning approach for distributed ransomware analysis," in *International Conference on Applied Cryptography and Network Security*. Springer, 2023, pp. 621–641.

[14] F. Ullah, G. Srivastava, S. Ullah, and L. Mostarda, "Privacy-preserving federated learning approach for distributed malware attacks with intermittent clients and image representation," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 4585–4596, 2023.

[15] G. Ciaramella, F. Martinelli, F. Mercaldo, C. Peluso, and A. Santone, "An approach for privacy-preserving mobile malware detection through federated machine learning," in *Proc. of 26th International Conference on Enterprise Information Systems*, 2024, pp. 553–563.

[16] Z. Çıplak, K. Yıldız, and Ş. Altınkaya, "Fedetect: A federated learning-based malware detection and classification using deep neural network algorithms," *Arabian Journal for Science and Engineering*, pp. 1–28, 2025.

[17] I. Baptista, S. Shiaeles, and N. Kolokotronis, "A novel malware detection system based on machine learning and binary visualization," in *2019 IEEE international conference on communications workshops (ICC workshops)*. IEEE, 2019, pp. 1–6.

[18] J. R. Rose, M. Swann, G. Bendiab, S. Shiaeles, and N. Kolokotronis, "Intrusion detection using network traffic profiling and machine learning for iot," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, 2021, pp. 409–415.

[19] F. Mercaldo and A. Santone, "Deep learning for image-based mobile malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 16, no. 2, pp. 157–171, 2020.

[20] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 2022, pp. 965–978.