Malware detection with Machine and Federated Learning

Guillermo Clara
Department of Computer Science
Georgia State University
Atlanta, Georgia
gclara1@student.gsu.edu

Abstract – Overtime Cyberattacks have become more common. Current detection techniques such as Signature-Based detection systems have shown to have certain disadvantages against new malware. This research focused on exploring the use of different Machine Learning models and approaches to predict whether an executable file is likely to be malware. Throughout the study, it was demonstrated that most models had acceptable performance metrics. With Federated Learning, it could be seen that the accuracy of the models can be high despite having a decentralization of data.

I. Introduction

Without a doubt, computers have become an essential part of human society and development. From personal computers, mobile devices to the Internet of Things, computers have revolutionized how people communicate and work. As computers increase their interconnection and usage of personal data, so does the risk of intrusion and malicious activities. Cyberattacks pose not only risks to people but also to businesses. In 2020, cyberattacks caused 1 trillion dollars in losses, 50% more than in 2018 [1]. Cyberattacks are defined as operational risks to information and technology assets affecting confidentiality, availability, and integrity of information systems [1]. While cyberattacks can include different techniques in different levels (e.g., application level or network level), this paper will focus on downloadable malware that exploit operating systems vulnerabilities.

Antivirus systems often use detection software to identify potentially dangerous malware. There are two approaches for detection, static and dynamic [2]. Static detection consists of analyzing the source code of the executable file. On the other hand, dynamic detection runs the malware on a safe isolated sandbox. The detection is done by comparing patterns in the analyzed file with others already found on a malware compilation database [2]. This is called Signature Based Detection. While effective, there are some limitations to this approach. Nowadays, these systems can be evaded with several techniques.

For example, some malware de-encrypt their code first and then execute, evading the detection systems [3]. There are approaches even more sophisticated such as obfuscation. A malware written so hard that it is difficult to analyze [3].

In recent years, Machine learning has become increasingly popular. Machine learning is a branch of artificial intelligence that focuses on using patterns and data to predict a certain outcome.

This research paper focuses on exploring how the implementation of different machine learning models fed with malware file patterns can be used to accurately identify potential malicious software.

II. Proposed method

The proposed idea is to implement machine learning powered malware detection into the operating system. The initial prediction model will be fed and trained with real world data about previously identified files containing malicious code as well as benign code. It is worth noting that the approach followed for detection is static, as this OS security system will not execute files to test them. Rather, it will extract different features of the file that can be used to categorize it or not as potential malicious software.

Since machine learning algorithms need to store large amounts of data, it is important to consider how and where to store the large datasets. Instead of just communicating with a centralized dataset, this research paper explores the approach to use federated learning to have a data-decentralized learning model.

A cloud server will host the global machine learning model which will initially deploy to the edge devices connected to it. Then, using their individual data, edge devices will train local machine learning models. Instead of sending datasets to the centralized server, edge devices will send their local models. Then, the centralized server will aggregate the models through federated averaging and create a new global model. This model will then be redeployed to the edge devices. This process would be repeated until the precision of the model reaches an acceptable level.

III. Study introduction

For this study, two different approaches of machine learning were compared to evaluate how different models can predict potential malware files. The study was performed in different phases. The first phase consisted of collecting real-world data. The second phase consisted of preprocessing the dataset to make it suitable for training. The third phase consisted in developing simulation software where the models can be tested with different settings and approaches.

IV. Methods

A. Data Collection

The data used for this research paper is structured in a tabular way. The dataset contains important features extracted from both malicious and benign files. The collection process began with malicious files. For security reasons, the extraction of the features from malicious code was not done in this study. Instead, data was collected from datasets provided by previous research [4]. The data pertaining to benign malware was collected through this study by parsing the executable files with Python. A total of 3800 benign files were parsed, most of them obtained from Win32 as well as already known benign third-party files (e.g., desktop applications)

The files used for this study use the Windows Portable Executable (PE) format (shown in Fig. 1). This format contains sets of information or modules that are used by the operating system for execution. Hackers target these headers so they can store and spread malicious content [5]. As a result, the different features found in the headers are essential to develop a machine learning model that can identify this kind of software. In fact, most of the previous research on the most important features found in PE files suggest including almost all the characteristics found in the headers. For this study, we included the PE Header features as well as the following sections: data, text, pdata, rdata, rsrc, and reloc.

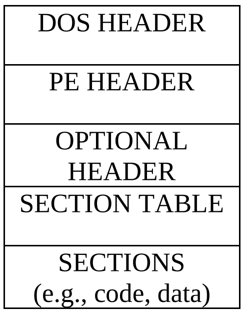

Fig 1. Different parts of a Windows Portable Executable All sections (except section table) were used for this study.

B. Data preprocessing

It is worth mentioning that the malware dataset had as a label the type of malware the file was. Since this problem's objective is to find out whether the file is malware or not, the dataset was modified to have a binary categorical feature representing type of software (i.e., Benign or Malicious).

Additional preprocessing techniques were applied. To normalize continuous features, a range normalization from 0 to 1 was applied. To reduce noise of data, IQR1.5 was used to clamp outliers.

Overall, the dataset had more than 100 features. However, having many features makes it more possible to have an overfitting model as a result [6]. Overfitting occurs when the prediction model "memorizes" the training set. As a result, while the prediction model works with test data from the original dataset, it is harder for it to work with new data such as real-world applications. To reduce this, feature selection was applied to the dataset. This was performed by using a Random Forest Classifier to create a prediction model and consecutively identify the features' importance score. The dataset features amount was reduced from 106 to 10 features (Shown in Fig. 2).

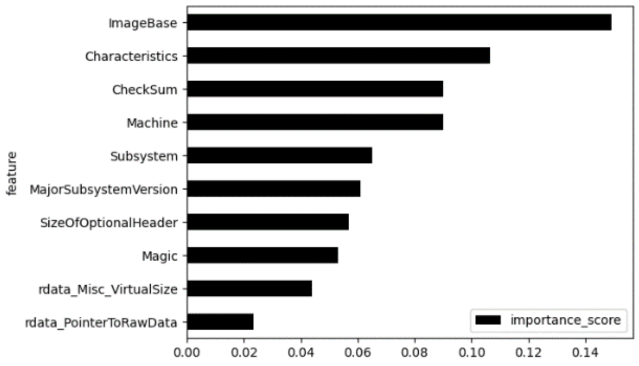

Fig 2. Top 10 most important features in the dataset.

C. Models Settings

One of the purposes of the study was to identify which machine learning model was the most accurate and efficient to be deployed. It is important to mention that some models were used in only one approach (i.e., centralized or federated). This is because the model's nature might not always be compatible with a federated approach. The study will explore the following models:

1. Gaussian Naïve Bayes: Probabilistic approach that assumes the distribution of data is gaussian (normal). Used in Centralized approach.
   - No special parameters used.

2. K-Nearest-Neighbors: Similarity based approach that classifies based on proximity. Used in Centralized approach. For results collection, the model was performed with 4 neighbors and using Manhattan distance.

3. Logistic Regression: Error-based approach derived from Linear Regression. Used for classification tasks. Used in both Centralized and Federated. For results collection the model was performed with max iterations of 100 for centralized approach and 10 for federated learning. Additionally, a learning rate of 0.002 was used.

4. Random Forest: Learning based approach. Ensemble of Decision Trees. Used in Centralized approach. Parameters used in the study consisted of number of estimators being 100, max depth being 5, and using Gini as criterion for importance score.

D. Federated Approach explanation in detail

The traditional way of using machine learning involves storing the data and models in a centralized server. However, this approach has some disadvantages. The high communication and storage costs, along with data privacy, will increasingly challenge the traditional eco-system of centralized over-the-cloud learning and processing for IoT platforms [7]. On the other hand, the federated learning approach aggregates only the models, not the data. This brings potential benefits such as preserving the privacy of user data, improving model performance, and having a more flexible scalability [7].

The federation process can be divided into 4 steps: A subset of contributing models is randomly selected from all the edge devices. Second, the global model is deployed from the server to the devices. Third, the devices train their models with their local data. Fourth, the devices send their trained models for aggregation in the central server. This process is repeated every iteration (also called federated round).

There are different approaches for model aggregation. One of the most popular techniques is called Federated Averaging. This approach consists of averaging the different local model parameters to set into the new global model every iteration [8]. For Logistic Regression, the chosen model for the study of this approach, aggregation consisted of weighted averaging of the models' coefficients and bias. The model's weight was determined by their local dataset size.

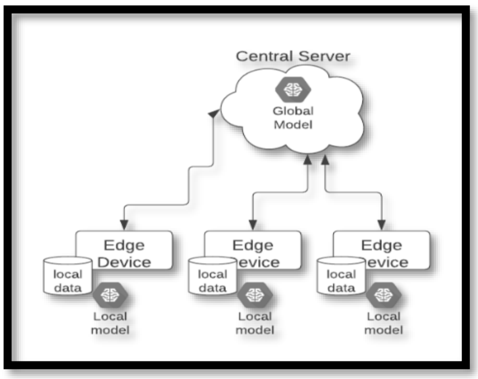

Fig 3. Basic view of Federated learning approach.

E. *Simulation Software*

The software was developed in Python. Users can interact with the program through the Command Line Interface. To use machine learning algorithms some libraries were needed (i.e., sklearn, NumPy, and pandas). Simulation takes two steps: Training and Testing.

On the training stage, the user can choose their desired machine learning model (e.g., Naïve Bayes) as well as their preferred approach (traditional or federated). Then, using sklearn, datasets are partitioned into training and test set. Sklearn then trains the machine learning model and shows the current model accuracy.

The testing phase consists of the user testing the performance of the machine learning model. They can either select from preselected known malware examples or they can also test their own files. The simulation also lets the user see the most optimal settings of their chosen machine learning model according to the previously set parameters.

The software also includes features to visualize the collection of various types of collected data (e.g., Visualizing metric improvement overtime in Federated Learning).

F. *Simulation Settings*

Since the data is also centralized on the Centralized approach, it was assumed that the amounts of Benign and Malicious samples were equally distributed. The number of used samples was 6000 (3000 malware samples and 3000 benign samples). The partition of training and test datasets always was 75% of dataset and 25% of dataset respectively.

As mentioned previously, Federated Learning is characterized by decentralization of data. As a result, it is fair to assume that most likely there will be edge devices with imbalanced data. To account for this, we decided to explore how the approach works in three different cases:

- Equal distribution (i.e., all edges have the same amount of data as well as the same proportion of malware/benign samples).

- Imbalanced distribution with balanced local malware/benign proportion

- Imbalanced distribution along with imbalanced local malware/benign samples

The specific settings used to generate the results for the three cases were number of edge devices = 20, learning rate=0.02, and number of iterations = 750. However, different executions of the software might give different results as the amount each device has is assigned randomly.

For this approach, we used 7% of the whole dataset as a testing dataset and 93% as the training set to be split among the edge devices. The metrics were calculated based on the performance of the same testing dataset.

G. *Evaluation Metrics*

The metrics used to evaluate the models (both centralized and federated) are the following:

- Accuracy: Measures how many predictions were correct.

- Recall: Measures how confident we can be that all the instances with the positive target level have been found by the model [9]. In the problem context, this means the proportion of actual malware correctly identified by the model.

- Precision: Measures how confident we can be that an instance predicted to have the positive target level has the positive target level [9]. In the problem context, this means the proportion of predicted malware that is malicious. This is an important metric to consider, as it can tell how prone the model is to predict false positives, which can be an inconvenience for the user and the system.

V. Results

TABLE I. Centralized models performance

Model/Metric	Accuracy	Recall	Precision
Gaussian Naïve Bayes	91.8%	97.4%	87.8%
K Nearest Neighbors	98.9%	98.7%	99.1%
Logistic Regression	95.3%	95.5%	95.1%
Random Forest	98.9%	99.3%	98.4%

According to Table I, the best performing models were Random Forest and K-Nearest Neighbors. While their accuracy performance was the same, Random Forest had higher recall and K-Nearest Neighbors had a higher precision.

As an additional exploration of the model's performance, we tested how both models classified a batch of benign files from different natures.

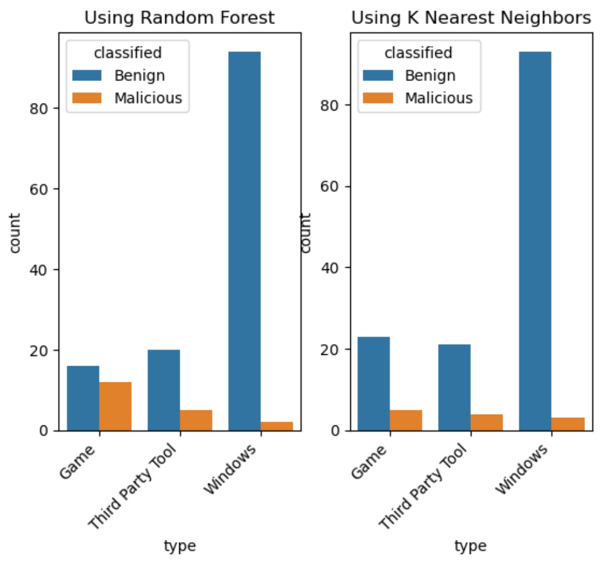

Fig. 4 Classification of benign files scanned with the software using Random Forest and K-Nearest Neighbors.

The other objective of the study was to explore how the federated learning approach could perform in terms of malware detection with machine learning. To generate these results, Logistic Regression was the only model used with this approach.

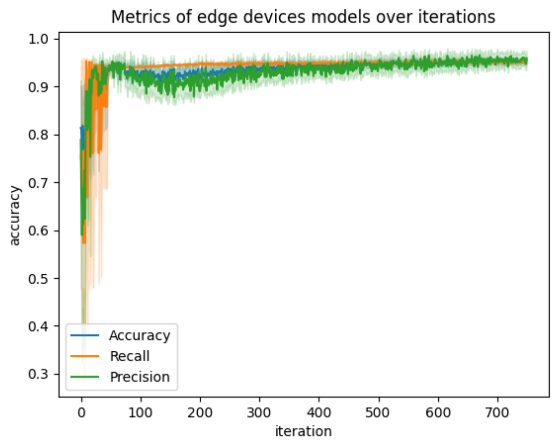

Fig 5. Metrics of edge devices with an even distribution and even proportion of malware and benign samples among local data.

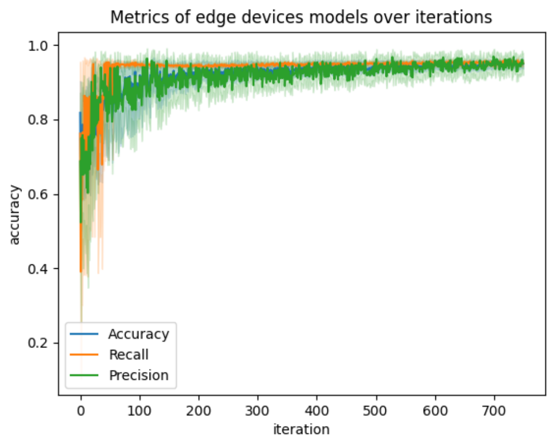

Fig 6. Metrics of edge devices with an even distribution but with disproportionate amounts of malware and benign samples among local data.

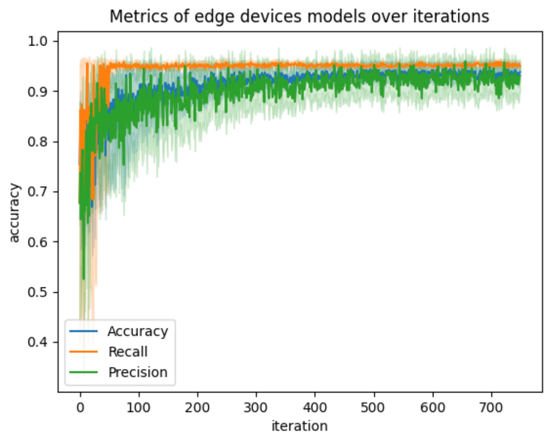

Fig 7. Metrics of edge devices with uneven distribution and disproportionate amounts of malware and benign samples among data.

VI. Discussion

Regarding the centralized approach, we can appreciate that the dataset works best with Information-based and Similarity-based machine learning models. In Fig. 4, we could appreciate that there seems to be a higher rate of misclassification among files related to games. On the other hand, Windows related files and third-party tools had low rates. This might indicate that files of a certain nature might have features similar to the ones found on malware.

The results on the federated logistic regression show that Federated Learning is a potentially good approach for malware detection. Comparing the different figures (5, 6, 7), we can see a pattern. As the distribution of the data gets more diverse among the edge devices, the iterations taken for the models to converge increase. Despite this, we could appreciate how the federated approach progressively improved the devices metrics and eventually stabilized them. This is an important observation as in a real-world scenario data is more likely to be unevenly distributed among devices as well as unevenly proportioned in terms of malware and benign samples.

VII. Conclusion

Based on the previous findings, we can conclude that both the centralized and the federated approaches have their advantages and limitations when it comes to malware detection. The centralized approach works best with Information and Similarity based learning models such as Random Forest and K-Nearest Neighbors. However, there are things yet to improve, such as adding many more samples of files of different nature (e.g., game-related files) to reduce misclassification.

The federated learning approach showed promising results for malware detection, indicating it can be a potentially good approach in the future. The study revealed a pattern in which the diversity of distribution and proportion of data increased the number of iterations required for convergence. Despite this, the approach gradually improved the devices metrics and stabilized them for all three presented cases, showing that the approach might be suitable for real-world application.

VIII. Future works

As mentioned before, both the centralized and federated approaches showed positive metrics in malware detection. However, there is still room for improvement. Some of the extensions of the research presented can include:

- Using the models on files from a greater variety of nature such as entertainment apps, videogames, and other high-performance programs to visualize better which areas are the most misclassified.
- Using larger amounts of samples of these natures to see whether misclassification is reduced.
- Exploring more models with different approaches such as Neural Networks.
- Using the models on known new malware files to see how effective the approach is beyond the dataset.

References

[1] F. Cremer, B. Sheehan, M. Fortmann, A. N. Kia, M. Mullins, F. Murphy, and S. Materne, “Cyber risk and cybersecurity: A systematic review of data availability,” *The Geneva Papers on Risk and Insurance - Issues and Practice*, vol. 47, no. 3, pp. 698–736, 2022.

[2] A. A. Elhadi, A. H. Osman, and M. A. Maarof, “Malware detection based on hybrid signature behaviour application programming interface call graph,” *American Journal of Applied Sciences*, vol. 9, no. 3, pp. 283–288, 2012.

[3] A. Silberschatz, P. B. Galvin, and G. Gagne, “Security,” in *Operating System Concepts*, 10th ed., Hoboken, New Jersey: Laurie Rosatone, 2018, pp. 633–634.

[4] M. I. Yousuf, I. Anwer, T. Shakir, M. Siddiqui, and M. Shahid, “A multi-feature dataset for Windows PE malware classification,” *SSRN Electronic Journal*, 2023.

[5] M. T. Kamble and Sridevi, “Feature extraction and analysis of portable executable malicious file,” *2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA)*, 2022.

[6] Chen, RC., Dewi, C., Huang, SW. *et al*. Selecting critical features for data classification based on machine learning methods. *J Big Data* **7**, 52 (2020). https://doi.org/10.1186/s40537-020-00327-4

[7] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, “Federated learning for the internet of things: Applications, challenges, and opportunities,” IEEE Internet of Things Magazine, vol. 5, no. 1, pp. 24–29, 2022.

[8] M. Hong, S.-K. Kang, and J.-H. Lee, “Weighted averaging federated learning based on example forgetting events in label Imbalanced Non-IID,” *Applied Sciences*, vol. 12, no. 12, p. 5806, 2022.

[9] J. D. Kelleher, “Evaluation,” in Fundamentals of machine learning for Predictive Data Analytics: Algorithms, worked examples, ... and case studies, MIT Press, 2020, pp. 476–476.