International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Sangraha360: An Unknown Malware Detection Framework with Federated Learning and Drift Detection

Sripooja Mallam[1], Gandewar Raja Balaji[2], Ginuga Vikas Reddy[3], Kichhannapally Tejaswi[4], Vishnu Deshmukh[5], Kailasa Bajrang[6], Dr. Rajasekaran Subramanian[7]

Submitted: 28/12/2023 **Revised**: 04/02/2024 **Accepted**: 12/02/2024

Abstract*:* Strong detection mechanisms are required due to the growing threat that malware poses to the security and integrity of digital systems. To improve malware detection systems, this research study investigates the relationship between Drift Detection and Federated Learning, with an emphasis on Android devices. The heterogeneity of the Android ecosystem, its vulnerability to different kinds of malware, and the ever-changing landscape of cyber threats pose formidable obstacles for researchers. The suggested method addresses the evolving strategies of malware by integrating drift detection to monitor real-time changes in data patterns. A decentralized paradigm called federated learning is applied to cooperative model training across various Android devices while maintaining user privacy. In this study, we introduce a framework where federated learning is used in a malware identification model for the first time, and it is strategically combined with Drift detection Algorithms

Keywords: *Malware Detection, Federated learning, Concept Drift, Drift Detection, Data Collection, Machine Learning.*

1. Introduction

The security and integrity of digital systems are increasingly threatened by malicious software or malware, so developing strong detection mechanisms is crucial for cybersecurity. The sophistication and diversity of malware, which ranges from conventional viruses to more clever variants like ransomware and advanced persistent threats, grows along with our reliance on digital technologies. The demand for efficient and flexible malware detection techniques has never been greater, given the persistent evolution of cybercriminals' strategies to evade established security protocols. In this research paper, the promising intersection of Drift Detection and Federated Learning in the field of malware detection is investigated. Our goal is to improve malware detection systems' effectiveness and responsiveness by merging these modern technologies where cyber threats are dynamic and evolving.

Although malware poses a threat across different digital environments, our research focuses specifically on malware for Android devices. Because of its open ecosystem and varied app store, Android is a popular platform for malware of all kinds, including spyware, ransomware, and trojan horses. The vulnerability of the platform is further increased by its fragmentation and the widespread use of custom ROMs and rooting. Being able to recognize unfamiliar Android malware is essential for creating preventative security measures and remaining ahead of the constantly changing threat landscape.

For researchers, developing efficient malware detection models for Android devices is a challenging task. It can be tough to create a single model that functions well with a variety of data sources

due to the diversity present in the Android ecosystem. It is more difficult to develop a model that can consistently identify malware throughout the entire system because of the variations in devices and apps. Given the sensitive nature of the data used in malware detection, protecting user privacy is crucial. Researchers must strictly adhere to ethical and legal requirements in order to guard against potential abuse and unauthorized access. Models must be able to adjust to new threats due to malware's dynamic nature, which is characterized by its constantly changing tactics and attributes.

Our research leads the way in using drift detection in conjunction with federated learning for malware identification to address these issues. We propose a framework named Sangrahra360. A strategic decision is made to combine drift detection with federated learning. Because federated learning is decentralized, it works well in the varied and dispersed Android ecosystem. Handling the variety of Android devices and maintaining individual privacy, enables us to train the model collaboratively on each device. A crucial component of our approach is drift detection, which tracks real-time changes in data. This is

[1] *Keshav Memorial Institute of Technology, Hyderabad,India ORCID ID : 0009-0000-7589-6911*
[2] *Keshav Memorial Institute of Technology, Hyderabad,India ORCID ID : 0009-0005-0235-2856*
[3] *Keshav Memorial Institute of Technology, Hyderabad,India ORCID ID : 0009-0007-7956-400X*
[4] *Keshav Memorial Institute of Technology, Hyderabad,India ORCID ID : 0009-0001-1528-3600*
[5] *Keshav Memorial Institute of Technology, Hyderabad,India ORCID ID : 0009-0003-4661-7938*
[6] *Keshav Memorial Institute of Technology, Hyderabad,India ORCID ID :*
[7] *Neil Gogte Institute of Technology, Hyderabad,India ORCID ID : 0000-0002-6572-3934*
Corresponding Author Email: rajasekarans@ngit.ac.in

essential as malware trends change. Our model can continuously adapt by identifying changes in data patterns, maintaining its performance in the face of new evolving malware samples

2. Literature Review

Much research has been done in the field of Android malware detection, and methods have been divided into two main groups according to the features used to predict malicious intent: Static and Dynamic. Historically, static analysis has been used extensively in the field to examine features without running the malware. Static Malware Analysis [1] [2]. Using Machine and Deep Learning [3] is one popular technique. This method uses cutting-edge machine and deep learning techniques to delve deeply into static features. Although it comprehensively analyzes these characteristics, obfuscated or polymorphic malware may undermine its efficiency.

Dynamic Malware Analysis Using Machine Learning Algorithm [4] is a notable example of dynamic malware analysis. Using machine learning algorithms, this technique executes malware samples in a controlled environment and examines their behavior while running [5]. Even though it's good at identifying patterns in behavior, sophisticated evasion techniques and zero-day attacks could pose problems.

The combination of machine learning and drift detection [6] is another significant development. This approach uses a combination of machine learning and drift detection mechanisms to adjust to changing malware patterns. Although it tackles the issue of concept drift [7] , real-time adaptability [8] may be impacted by the complexity involved, making the drift detection method selection essential.

Federated Learning with FedAvg[9] presents a decentralized paradigm in which users train models locally on their devices, and local updates are aggregated to create a global model. This strategy protects the privacy of data and shows resilience to centralized attacks. Nevertheless, there are drawbacks, including potential problems with heterogeneous datasets and communication overhead. Federated Learning with FedAvgM improves the federated learning framework by incorporating adaptive model averaging, which leads to better convergence in non-IID scenarios, building upon FedAvg . The non-IID data distribution problems are addressed by this adaptation, but for best results, hyperparameters must be carefully adjusted.

The emphasis switches to managing distributed concepts [10] and drift in a federated learning environment in the context of Federated Learning under Distributed Concept Drift [9]. This method introduces mechanisms that allow models to adjust to changing conditions across multiple devices; however, its efficacy depends on its ability to detect distributed concept drift accurately and respond appropriately.

Federated Learning with Dynamic Weighted FedAvg and Drift Detection is a novel approach that combines drift detection and dynamic model averaging with the concepts of federated learning. The goal of this framework is to get around static analysis's drawbacks, especially when it comes to obfuscated malware. Moreover, it tackles issues brought about by changing malware patterns and dynamic environments, providing a more flexible and reliable solution.

In conclusion, while each technique adds to the developing field of Android malware detection, the suggested framework aims to combine the benefits of drift detection, dynamic approaches, and federated learning for improved adaptability and privacy-preserving qualities.

3. Preamble

This section provides a quick overview of the subjects covered in the research process, including frameworks, federated learning, and datasets

3.1. FLOWER FRAMEWORK

"Federated Learning with Overlapping Experiences," or "Flower," [11] offers a robust and adaptable framework. Infrastructure is needed for federated learning, aggregation, and analytics in order to transfer machine learning models back and forth, train and compute them on local data, and then compile the modified models. Using the Flower framework any workload, any ML framework, and any programming language can be federated by the user.

This section explores the importance of the Flower framework, providing details on its features, architecture, and critical role in developing federated learning environments

FLOWER ARCHITECTURE

The client-server architecture of the Flower framework is intended to effectively coordinate the federated learning process. Flower is primarily made up of two parts: the clients and the server.

1) server: The Flower architecture is controlled by the server. It is in charge of directing and coordinating the federated learning process. Like a conductor in an orchestra, the server synchronizes the actions of multiple devices, or clients, to produce an accurate result. Its principal duties consist of:

● **Global Model Management**: The global machine learning model, which forms the basis of the federated learning procedure, is maintained by the server. Initially, this global model is developed and made available to each client..

● **Managing Training Rounds**: Flower divides the federated learning process into rounds and works in a cyclical manner. The server assigns the global model to the clients for individual local training in each round.

● **Aggregation of Model Updates**: The server gathers and aggregates each client's model updates after the clients have finished their local training with their individual datasets. The knowledge gathered from various local datasets is integrated through this aggregation process, which is essential for enhancing the global model

2) Client: Within the Flower framework, clients stand in for the decentralized nodes or gadgets that take part in federated learning. It is the client's responsibility to use its own dataset for local model training. An explanation of the client's role is provided here:

● **Local Model Training**: The entities that own local datasets are referred to as clients. Using these datasets, they train models to identify patterns and insights that are specific to the data. The global model is guaranteed to gain knowledge from the different backgrounds of all participating clients thanks to this local training procedure

● **Model Update Submission**: Using its dataset, each client creates a model update following local training. This update includes details on how to improve the model using the insights from the local training.

● **Communication with the Server**: In order to exchange information, clients communicate with the server. The server receives their updates for the model, aggregates them, and improves the global model. A crucial component of federated learning is communication, which Flower streamlines for maximum effectiveness

3.2. DATASETS

● In our study, the efficacy and generalizability of the suggested federated learning model for Android malware identification are significantly influenced by the dataset selection. There are many good datasets out there such as DREBIN [12] , AndroZoo [13] , MalImg [14] , KronoDroid [15] etc. The KronoDroid dataset is one of the most important datasets for our research.

● KronoDroid offers an extensive and varied selection of Android apps. It is a carefully selected dataset designed for Android malware research. This dataset contains a diverse array of malware samples, each displaying distinct traits and actions.

● The characteristics of the Kronodroid dataset are:

1. Data with labels (malware/benign samples)

2. 289 system calls, or dynamic features

3. 300 static features (metadata, permissions, intent filters)

4. four different timestamps for each data sample,

5. covers the years 2008–2020 in Android history

4. Concept Drift

The phenomenon known as "concept drift" describes how a target variable's statistical characteristics, which a machine learning model seeks to predict, might alter over time. Concept drift can happen in datasets when there is a change or evolution in the relationship between the target variable and the input features. Dealing with Concept drift is very crucial in malware prediction due to its evolving nature.

One major difficulty is the dynamic nature of malware. Malware features change over time as attackers create new evasion strategies, which makes it harder for models trained on historical data to recognize previously undiscovered variations. The emergence of new malware strains has the potential to disrupt the class balance within a dataset resulting in data imbalance. In order to maintain effectiveness in recognizing both known and new malware, concept drift handling is crucial as traditional models may become biased towards the majority class.

A potential consequence of neglecting to address concept drift is a reduction in model performance. Maintaining the model's accuracy over time requires regular monitoring and updating based on incoming data, especially when it comes to identifying novel malware strains. The key to cybersecurity is real-time detection. Concept drift handling guarantees that models continue to be sensitive to new threats, allowing for the prompt detection and reduction of security problems.

Concept Drift Visualization in a dataset is done using some statistical tests. In our attempt to visualize Concept drift in the KronoDroid dataset chi square statistical test is used. Using the chi square statistical test features with Concept drift are known. A graph is plotted for 5 of those features out of which 4 features have drift and one feature doesn't have drift. The graph is plotted between feature values and data samples. Inconsistencies in the drift features and the consistency in the non-drift features are clearly observed in Fig 1.

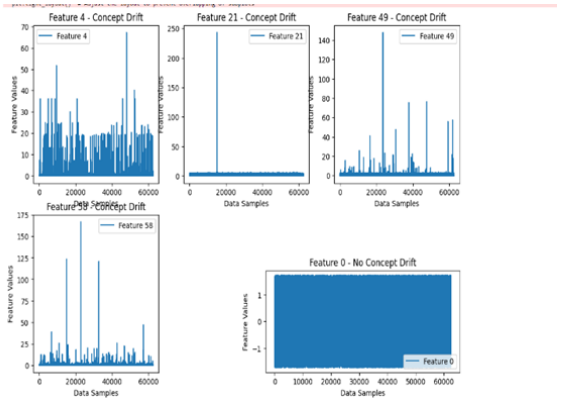

Fig. 1. Visualization of difference between drift features and Non-drift features in Kronodroid dataset

5. Methodology

In this section, the methodology employed in our research is outlined, detailing the integration of federated learning for Android malware identification. A comprehensive explanation of our approach to data collection, aggregation techniques for federated learning, and the associated metrics is provided

5.1. DATA COLLECTION

An Android application is used to carry out the data retrieval process, which makes use of a number of functions provided by the Android SDK to obtain the necessary data for model training and prediction. Users can choose an installed application from a dropdown menu displaying all installed applications in the application interface. Both static and dynamic data collection are made easier by the application. At predetermined intervals, dynamic data about the Android device and static data unique to the selected application are collected.

In addition to being updated for user reference on the display, the collected data is also sent to a server for safe storage in a global database. This dual functionality actively involves every application user in the ongoing data collection process while also improving the model's capabilities. This application makes use of Django to handle the backend in an effective manner. MongoDB offers secure storage for the data. The integration of the Android SDK, Django, and MongoDB highlights a holistic approach to data storage and retrieval, enhancing user experience and supporting ongoing model enhancement.

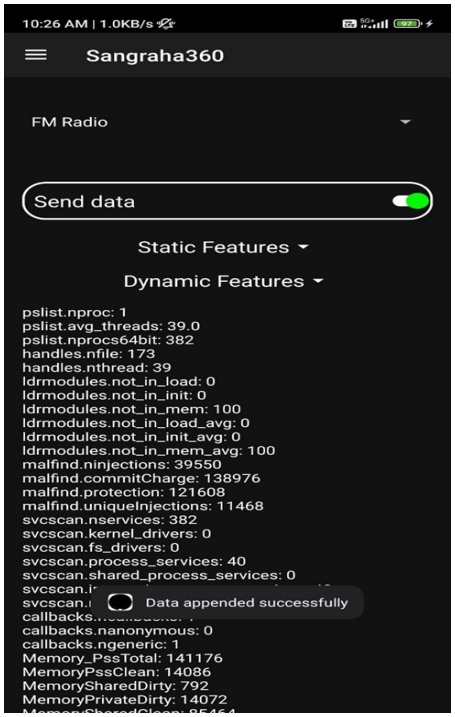

Fig 2 : Static and Dynamic features data collection android app

5.2. PROPOSED ALGORITHMS

5.2.1. Federated Aggregation Algorithms

Federated learning algorithms facilitate the collaborative training of models across these decentralized entities.

There are several types of Federated Learning algorithms, which have their characteristics but in consideration of the model and the goal for the model, two algorithms are chosen.

5.2.1.1. *Federated Average (FedAvg):*

A global model is initialized on a central server, and the local models are trained on clients using their local data. The local model's parameters are then sent to the central server, which computes the average of all the parameters to update the global model.

The FedAvg [9] method is chosen as it has many benefits which align with our ideal model. It reduces the amount of raw data that must be transferred, which reduces the amount of data that must travel over the network. Emphasizing the important model changes instead of all user data significantly reduces communication overhead and encourages more effective data sharing within the federated learning framework.

Furthermore, the concept of enhanced generalization in federated learning is predicated on the utilization of diverse data sources. The model is exposed to a wide range of data types during training since it is trained on datasets derived from different devices. Because of its diversity, the model is better able to generalize and recognize patterns in malware that have never been observed before. The model's potential is increased by adding knowledge from other sources, which increases the model's flexibility and resilience.

The main flaw in FedAvg is that it can cause delays or slower computation completion for stragglers or slow learners in a federated network. This could rely on a number of factors, including processing power, latency in the network, additional data, model complexity, and synchronization problems. A novel approach known as "Dynamic Weighted Federated Average" (DW-FedAvg) was developed to address this problem.

5.2.1.2. *Dynamic-Weighted FedAvg (DW-FedAvg):*

A method called Dynamic Weighted Federated Average (DW-FedAvg) [6] aims to improve federated learning systems' performance. All clients are treated equally in traditional federated averaging, provided that their models are of comparable quality and their data is equally representative. This presumption might not apply in practical situations, though. This strategy's main concept is to dynamically assign varying weights, with an emphasis on accuracy, to each client's model updates based on their performance metrics.

The accuracy of each client's local model in relation to past metrics is used to assess how well they performed. The chosen metric serves as a gauge for how well or poorly the local model is doing. By modifying the weights allotted to each client and their contribution to the overall model, dynamic weighting is put into practice. More weight is awarded to clients who perform better, as demonstrated by increased accuracy or improvement. This indicates that the global model is more affected by their metric. Conversely, less valuable clients are given a lower weight, which lessens their influence on the overall model.

DW-FedAvg stands out for its versatility across several federated learning cycles. Every round, the weights allotted to the clients are dynamically adjusted, enabling the system to adjust to variations in the distribution of client data, model quality, or other elements influencing each client's relative contribution. DW-FedAvg offers a lot of benefits. By giving customer feedback priority over inferior models, it seeks to increase model quality. Additionally, the approach is resilient to the natural heterogeneity of clients in learning environments and easily adjusts to various data distributions and model attributes.Furthermore, DW-FedAvg is adaptable, which is vital in situations where clients and data may change over time.

In federated learning systems, the strategy may lower overall communication costs by emphasizing more valuable and efficient customer updates. Additionally, it is highly effective against model poisoning, a growing concern in the context of federated learning models.

5.2.2. **Drift Detection Algorithms**

From [16] is where the Drift detection techniques used in our approach were taken . The purpose of these techniques is to precisely tackle the problems caused by idea drift in Federated Learning (FL) environments, where heterogeneous data is present across multiple clients and over time.

5.2.2.1. ***FedDrift-Eager***

This algorithm deals with situations in which there is only one new idea that appears at once. For each client, it integrates a local drift detection algorithm to detect the emergence of a new concept. When a client recognises a novel idea, they divide it into a new cluster and initialize a new model for group training. For staggered drift patterns where only one concept changes at a time this algorithm works well.

To explain the Drift detection algorithm in detail, if the model's loss over newly arrived data ($\ell(\tau)c,m$) degrades by a threshold δ relative to the loss at time $\tau-1$, it indicates a drift at client c and time τ. This test identifies any drift that degrades performance, but the requirement for building a new model looks specifically for concept drifts that match an unobserved and incompatible concept for every model that already exists. If the current model's minimum loss is greater than the previous model's minimum loss plus δ, a new model is generated.

It however encounters difficulties, when two or more concepts appear at the same time. It attempts to train a single model for both new concepts using a single cluster, which is a suboptimal solution. In order to overcome this constraint, the algorithm is expanded to manage the general scenario in which an arbitrary quantity of novel ideas may emerge concurrently.

5.2.2.2. ***FedDrift***

This algorithm deals with the general scenario wherein several new malware trends could appear at the same time. To conservatively isolate clients by detecting drift in individual clusters, it makes use of hierarchical agglomerative clustering. Clusters corresponding to the same concept are gradually merged over time, allowing for a steady and secure adaptation to changing concepts. The algorithm is flexible enough to handle a wide range of drift patterns and can be tailored to an unknown number of new malware trends that might emerge in the future.

The approach makes use of a general hierarchical clustering process that includes a stopping condition and a distance function over the collection of elements to be clustered. FedDrift collects loss estimates of models assessed over a subsample of data linked to the cluster for every model in order to define a distance function. The initialization of cluster distances is determined by the variation in loss estimates, which gauges how much a model's accuracy deteriorates when applied to its own data.In contrast to FedDrift-Eager, it requires more computational resources, generates a greater number of global models (M), and introduces an O(M2 log M) time complexity at the server for each time step. The algorithm's efficiency can be optimized by creating fewer overall models. Clients can also maintain relevant data and weights for a sliding window of recent time steps to facilitate subsampling steps.

5.3. Results And Conclusion

METRICS: metrics are quantitative measures used to evaluate the performance and effectiveness of a model in solving a particular task. These metrics provide insights into how well the model generalizes to new, unseen data and how accurately it makes predictions. Different types of machine learning tasks, such as classification, regression, clustering, and more, often require specific metrics tailored to the nature of the task

Accuracy: It is calculated as the ratio of the number of correct predictions to the total number of predictions

Loss: It is calculated as the average of the losses of all clients.

F1 Score: It is calculated as the harmonic mean of precision and recall.

Precision: It is calculated as the ratio of true positives to true positives plus false positives. It is the proportion of positive samples that are correctly classified as positive.

Recall: It is calculated as the ratio of true positives to true positives plus false negatives. It is the proportion of positive samples that are correctly classified as positive.

AUC (Area Under the ROC Curve): It is a measure of how well a binary classification model can distinguish between positive and negative classes. AUC provides an aggregate measure of performance across all possible classification thresholds.

FPR (False Positive Rate): It is the ratio of false positives to the total number of actual negatives.

Table 1

Metrics	FedAvg	DW-FedAvg
Accuracy	0.93253	0.96009
Loss	0.410637	0.420644
F1 Score	0.963261	0.978453
Precision	0.978715	0.985501
AUC	0.921036	0.971521
False Positive Rate	0.051538	0.028479

Table 1: Federated learning aggregation results with Kronodroid dataset

Table 2

Metrics	FedDrift-Eager	FedDrift
Accuracy	0.9552	0.9849
Loss	0.4543	0.4691

Table 2: Drift detection results for the Kronodroid dataset with federated learning

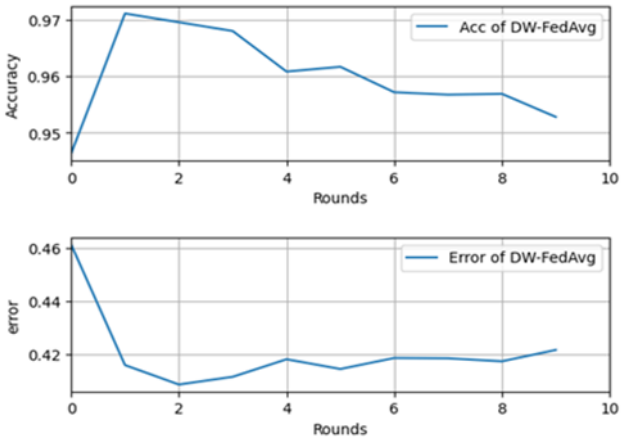

Fig 3: Graphs of Accuracy and Error rate values using DW-FedAvg

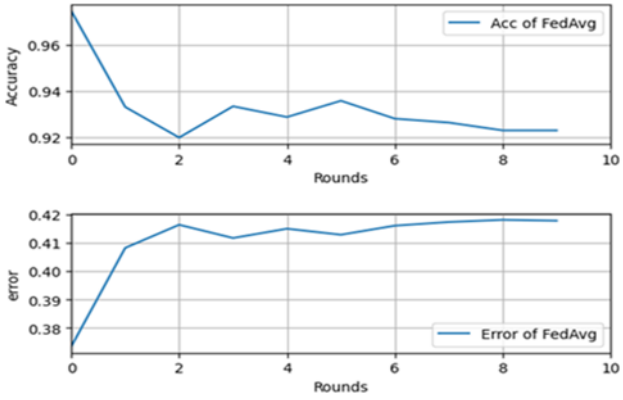

Fig 4: Graphs of Accuracy and Error rate values using FedAvg

The model is performing extremely well when DW-FedAvg aggregation is being used and when the FedDrift algorithm is being used. In terms of accuracy and error rate as seen from the table and graphs with DW-FedAvg we have an accuracy of 0.96009 and with FedAvg we have an accuracy of 0.93253. The same way with FedDrift we have an accuracy of 0.9849 and with FedDrift -Eager we have an accuracy of 0.9552.This clearly shows us that with these aggregation and drift detection algorithms this model outperforms a lot of traditional models out there

6. Effective Implementations

To apply federated learning in the real-world an application[11] is used. This application can be downloaded onto the client's Android device. It will be sent the global model from the server and the local training starts on the device using the device data. After the training is done the device is equipped with an application for malware identification service which uses federated learning. For the server to connect to real time clients all the input it needs is port number and IP address. When provided with that information the client gets connected and can participate in training and predictions.

All the research information till now and future findings and

improvements will be available at the website "http://sangraha360.org" . This website contains all the applications, codebases and theory. To know all the future upgrades and improvements this website can be visited .

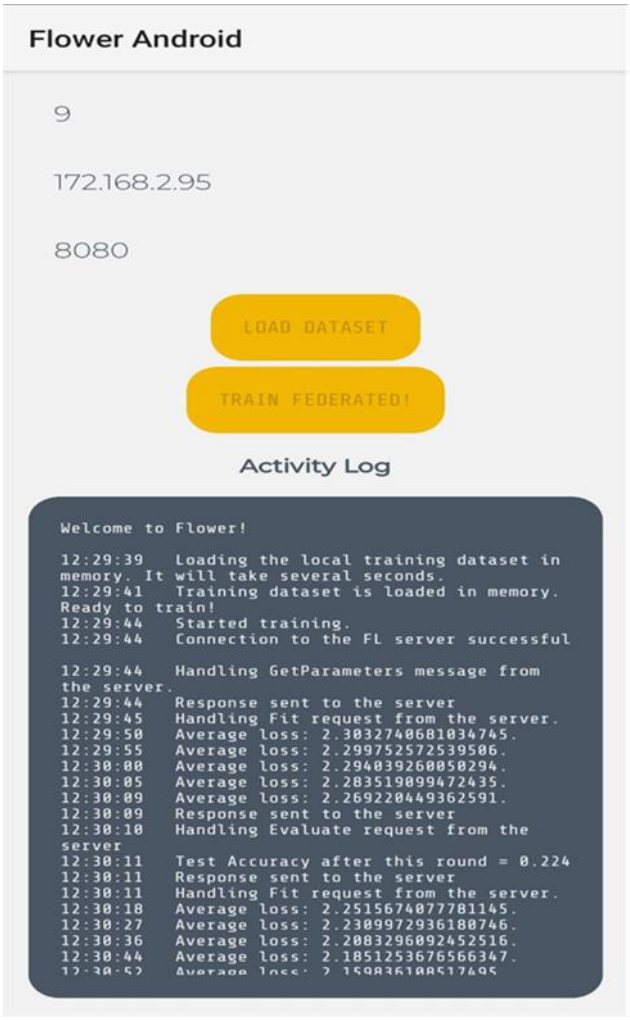

Fig 5: Federated learning application for malware detection [7]

Acknowledgements

Mr.Stephen Lobo, President, and Co-Founder, CyberGuard360 Inc, USA for Research Guidance. Prof Neil Gogte, Director , Keshav Memorial Institute of Technology for the Project Guidance, Finance and Material Support

Author contributions

Sripooja Mallam: Literature survey, analysis of limitations in previously proposed research work and proposal of new approach with drift detection and active learning

Balaji Gandewar: Comprehensive contributions across research, development, and integration of an Android application including data retrieval and analysis features, backend integration for periodic data collection, auto-update, malware detection, user registration, token authentication, and end-to-end testing to validate server functionality. Ensured code cleanliness according to Kotlin conventions with detailed documentation comments.

Bajrang Kailasa: Retrieved static features using Kotlin libraries, conducted comprehensive analysis of data dumps generated by various tools, and utilized Android profilers to analyze app performance ,also implemented procedures for generating data dumps from different apps using ADB tools, Python scripts for parsing data dumps, and automated data dump generation processes.

Vikas Reddy: Development and implementation of the federated learning approach, integrating the Flower framework and various aggregation algorithms. Conducted experiments, analyzed results, and contributed to the performance evaluation and interpretation of findings.

Kichhannapally Tejaswi: Worked on Drift Identification and Visualization in various datasets.Development and implementation of the federated learning approach,with fedAvg aggregation technique.Conducted experiments, analyzed results, and contributed to the performance evaluation by adding drift detection algorithms to federated learning environment

Vishnu Deshmukh:The backend development for the project involved the creation of a dedicated endpoint to integrate the Android app UI with the backend logic. This included coding to handle and process results derived from static and dynamic features. To ensure efficient data management, integrated MongoDB to store both static and dynamic features securely. Contributing to the project's security, implemented token authentication, restricting access to whitelisted users only. Additionally, incorporated Firebase for the secure storage and management of authentication tokens, reinforcing the overall security of the system and limiting access to protected endpoints.

Dr.Rajasekaran Subramanian: Lead the Research, Guidance and reviewed the publication draft.

Conflicts of interest

The authors declare no conflicts of interest

References

[1] Himanshu Kumar Singh, Jyoti Prakash Singh "Static Malware Analysis Using Machine and Deep Learning"

[2] Y. Pan, X. Ge, C. Fang and Y. Fan, "A Systematic Literature Review of Android Malware Detection Using Static Analysis,"

[3] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Aguera y Arcas "Communication-Efficient Learning of Deep Networks from Decentralized Data".

[4] Udayakumar .N, S Anandaselvi, Dr T Subbulakshmi "Dynamic malware analysis using machine learning algorithm"

[5] Fabricio Ceschin, Marcus Botacin, Heitor Murilo

Gomes, Felipe Pinage, Luiz S. Oliveira, Andre Gregio "Fast & Furious : Modelling Malware Detection as Evolving Data Streams".

[6] E Ayushi Chaudhuri, Arijit Nandi, Buddhadeb Pradhan "A Dynamic Weighted Federated Learning for Android Malware Classification"

[7] Jordaney, R., Sharad, K., Dash, S. K., Wang, Z., Papini, D., Nouretdinov, I., & Cavallaro, L. (2017). "Transcend: Detecting concept drift in malware classification models"

[8] Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., & Cavallaro, L. (2018). TESSERACT: eliminating experimental bias in malware classification across space and time

[9] Sashank Reddi , Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konecny, Sanjiv Kumar, H. Brendan McMahan "Adaptive Federated Optimization"

[10] Anderson, H. S., Kharkar, A., Filar, B., Evans, D., & Roth, P. (2018). Learning to evade static pe machine learning malware models via reinforcement learning

[11] Beutel, Daniel J and Topal, Tanner and Mathur, Akhil and Qui, Xinchi and Fer "Flower: A Friendly Federated Learning Research Framework"

[12] Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., & Rieck, K. (2014). "Drebin: Effective and explainable detection of android malware in your pocket"

[13] Allix, K., Bissyand´e, T. F., Klein, J., & Le Traon, Y. (2016). Androzoo: "Collecting millions of android apps for the research community"

[14] Nataraj, L., Karthikeyan, S., Jacob, G. and Manjunath, B., 2011. [online] dropbox.com. Available at: "[Accessed"

[15] Alejandro Guerra-Manzanares, Hayretdin Bahsi, Sven Nõmm,"KronoDroid: Time-based Hybrid-featured Dataset for Effective Android Malware Detection and Characterization"

[16] Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, Phillip B. Gibbons " Federated Learning under Distributed Concept Drift"