RESEARCH ARTICLE

DOI: 10.34256/ijcci2423

International Journal of Computer Communication and Informatics

Detecting and Classifying Malware in Network Traffic

K.P. Akshithaa Parvathavardhini [a,*], P.K. Nizar Banu [a]

[a] Department of Computer Science CHRIST (Deemed to be University), Bangalore, Karnataka 560029, India

[*] Corresponding Author: akshithaapv26@gmail.com

Received: 01-09-2024, Revised: 29-11-2024, Accepted: 08-12-2024, Published: 22-12-2024

Abstract: The range of threats of Android malware has grown due to the widespread use of Android devices. The intricacy and diversity of malware is always changing, making it difficult for conventional signature-based detection techniques to stay current. In this regard, network traffic analysis is a viable method for identifying and categorizing Android malware. Modern cybersecurity plans now include detecting and categorizing malware in network traffic as a vital component, given the increasing reliance of organizations for communication and data exchange over interconnected networks. The foundation of modern society is now the interchange of information over networked systems. The integrity and security of networked systems are seriously jeopardized by malware, which is one of the many cybersecurity dangers that digital infrastructures are exposed to due to their interconnection.

Keywords: Android malware, Malware analysis, Malware category, Malware family, Classification, API call, Cyber threat, Behavioural analysis.

1. Introduction

In recent years, mobile devices have become indispensable, easily blending into our daily lives for everything from online banking and shopping to social communication and leisure. However, as people rely more and more on mobile devices, mobile security dangers have increased. Mobile malware, which includes viruses, worms, trojans, and other harmful software, has grown in size and sophistication, providing substantial issues for both individuals and organizations. These dangers can have disastrous repercussions, such as data breaches, financial losses, and compromised user privacy, emphasizing the crucial need for effective detection and protection systems. The huge number and diversity of

malware kinds, as well as their rapid growth, make mobile malware detection even more challenging. Traditional detection methods, such as signature-based approaches, have grown ineffective over time, particularly against unknown or polymorphic malware that constantly mutates to avoid detection. These methods are based on predetermined patterns of known malware, making them incapable of detecting new or zero-day threats. As a result, there is a high demand for innovative and adaptive solutions that can keep up with the changing nature of malware [1]. Machine learning (ML) has emerged as a viable strategy for addressing this issue. By leveraging data analysis and pattern recognition, ML-based systems may discover and categorise malware more efficiently and correctly than traditional methods. Unlike signature-based approaches, ML models can evaluate massive volumes of data, find hidden patterns, and forecast previously unknown malware by analysing system behaviour, network activity, and file properties. This versatility makes machine learning a great tool for improving mobile security. Android is the world's most popular mobile operating system. Its popularity, however, has made it a prominent target for cybercriminals, who take adssvantage of its open-source nature and broad use to create and transmit malware [2], [3]. As a result, detecting and combating Android-specific malware hsas become a primary concern for cybersecurity researchers.

This study investigates the use of machine learning approaches to detect and categorise Android malware. It investigates a variety of machine learning methods, such as Logistic Regression, Random Forest, Decision Trees, Support Vector Machines (SVM), and K-Nearest Neighbours (KNN), and evaluates their efficacy in detecting malware from a dataset that includes both benign and malicious samples. This study seeks to determine the best successful ways for Android malware detection and classification by analysing various algorithms in terms of accuracy, precision, recall, and computing economy. In doing so, it hopes to help build more robust and scalable solutions for safeguarding mobile devices in an increasingly connected world.

2. Related Work

Audry *et al.* (2023) investigate the application of machine learning to improve Android malware detection by analyzing system calls, which are crucial for app operations and provide valuable insights into app behavior. The researchers point out the limits of typical feature selection approaches, which frequently include noisy features that reduce detection accuracy. To address this, they recommend binary scoring of system calls in conjunction with advanced feature pickers such as ML, GSS, and DFS. A dataset containing malware and trusted app traces was analyzed using machine learning methods such as Random Forest, Support Vector Machine, Decision Tree, Naive Bayes, XGBoost, K-Nearest Neighbours, and Adaptive Boosting.The Decision Tree model performed admirably, with the highest accuracy (95.03%). However, its efficiency is influenced by dataset bias, and its applicability to developing malware strains requires further testing. Future research should investigate hybrid models that combine deep learning and real-time detection techniques for practical use in network security systems. Performance measurements such as precision, recall, and F1 score are used to validate the models' performance. The study emphasizes the need to select appropriate algorithms for effective malware detection, as well as the function of system call analysis in improving Android smartphone security (Audry et al., 2023) [4].

El Fiky *et al.* (2022) discuss the growing threat of Android malware, which increasingly targets important applications such as healthcare, banking, and e-Commerce. To address the difficulty of accurate malware detection and classification, the authors offer a parallel machine-learning model that can dynamically analyze and identify both categories and families of Android malware. Using the CCCS-CIC-AndMal2020 dataset, which contains 14 main malware categories and 180 important malware families, the model performs excellently through feature selection and machine learning classifiers. The trials show a category classification accuracy of over 96 percent and a family classification accuracy of more than 99 percent, beating previous approaches. The suggested approach is highly scalable and suitable for analyzing large datasets, providing multi-class characterization with high precision and low false positive rates. Future ideas include creating an online service that allows users to detect malware and classify it by category and family before installation, considerably improving Android smartphone security (El Fiky et al., 2022) [5].

Sethi *et al.* (2022) present a novel malware analysis paradigm to address the increasing complexity and diversity of malware, rendering existing signature-based detection methods ineffective. The framework uses Cuckoo Sandbox to perform static and dynamic analyses to extract information like as API calls, imported libraries, and existing signatures, which are then utilised to create a comprehensive dataset. The researchers used a two-level classification approach: the "Macro" classifier finds malware, while the "Micro" classifier recognises specific malware types like Trojan, Spyware, or Adware [6, 7].

The study uses the Weka framework to analyse many machine learning algorithms, including J48 Decision Tree, SMO (Sequential Minimal Optimisation), and Random Forest. The results show that J48 Decision Tree outperformed the other models, obtaining 100% accuracy in both detection and classification tests. SMO and Random Forest likewise had high detection rates, with 99% and 97%, respectively, while classification accuracies were 91% and 66%.

The authors acknowledge limitations due to the short sample size (220 files), and intend to increase the dataset in future work to incorporate a broader range of malware traits. They also want to use feature selection approaches to optimize the feature set and enhance model performance on larger datasets (Sethi *et al.*, 2022) [8].

Narudin *et al.* (2014) use an anomaly-based approach to evaluate machine learning classifiers for mobile malware detection, leveraging network traffic-derived characteristics. They use the MalGenome and self-collected datasets to compare classifiers like Random Forest, K-Nearest Neighbours (KNN), Decision Tree (J48), Multi-Layer Perceptron (MLP), and Bayes Network. Random Forest has the maximum detection accuracy of 99.99%, while KNN has the highest true-positive rate of 84.57% when detecting the most recent Android malware [9]. The paper emphasises the importance of feature selection and suggests future research into real-time, cloud-based detection systems that use machine learning, game theory, and multi-agent systems to improve adaptability and robustness. It also emphasises the need to investigate the correlations between features in order to increase detection accuracy and efficiency [9]. Aslan and Yilmaz (2021) [10] suggested a novel hybrid framework for malware classification that uses deep learning approaches to overcome the limitations of standard malware detection methods. With the increasing reliance on computer systems and the Internet, as well as the surge in cyberattacks, malware detection has emerged as a critical part of cybersecurity. Traditional detection methods, such as signature-based and heuristic-based approaches, struggle to detect newer malware variants due to enhanced obfuscation and packaging tactics [11]. To address these issues, the authors proposed a hybrid architecture based on deep learning that improves malware detection and classification accuracy [12].

The approach use a combination of transfer learning and pre-trained deep learning models to efficiently extract highlevel characteristics from malware samples converted to greyscale photos. The model combines two comprehensive pre-trained network topologies to improve feature extraction and drastically minimise the feature space. The suggested architecture was extensively tested on three widely recognised malware datasets: Malimg, Microsoft BIG 2015, and Malevis. The model's capacity to classify various malware families with high precision, recall, and accuracy was proved by its performance on the Malimg dataset, which reached 97.78%. This performance outperforms several of the most advanced approaches described in the literature. The study emphasises the value of deep learning in solving the challenges of current virus detection. Key contributions include the unique hybrid architecture, efficient feature extraction, and significant decrease in feature space, all of which improve the system's accuracy and performance [1]. However, the authors identified shortcomings in recognising specific malware variants that used advanced obfuscation techniques. They proposed additional study to develop detection algorithms that especially target such difficult malware strains.

The study emphasizes the potential of deep learning, particularly hybrid techniques, as a viable route for addressing increasing cybersecurity risks by providing a robust solution for malware classification.

3. Methodology

3.1 dataset info

The CCCS-CIC-AndMal-2020[13] dataset is an academic Android malware classification dataset created by the Canadian Institute for Cybersecurity (CIC) in collaboration with the Canadian Centre for Cybersecurity (CCCS). It is designed to aid research in Android malware detection and classification.

3.2 Dataset Description

Purpose: The dataset is intended for use in the research and development of machine learning models and algorithms for the detection and classification of Android malware. Authors:

The dataset was created by David Sean Keyes, Beiqi Li BsC., Dr. Gurdip Kaur, Dr. Arash Habibi Lashkari, Dr. Francois Gagnon, Dr. Fred ´ eric Massicotte, Abir Rahali MsC., and Laya Taheri MsC. Credit goes to these original authors for compiling and publishing the dataset. Citation: Users of the dataset are encouraged to cite the original papers (references 1 and 2) associated with the dataset to give credit to the authors and acknowledge the source of the data. Versions: V1: The base dataset is provided in CSV format. It can be downloaded from a specified source. V2: This version involves cleaning and organizing the dataset. The data is stored in Parquet files, which are more efficient for handling large datasets. The features are derived from both dynamic analysis (behavioral features) and static analysis (structural features) of Android applications. In V2, all data types are correctly set, and there are no missing records, ensuring that the dataset is clean and ready for analysis. Contents: The dataset contains features extracted from both dynamic and static analysis of Android applications. These features are essential for training machine learning models to classify applications as either benign or malicious.

Here are six characteristics extracted to understand the behavioral changes of these Android malware categories and families. The following are the significant characteristics that were extracted:

Memory Features: It defines the activities performed by malware utilizing memory. (23 Features)

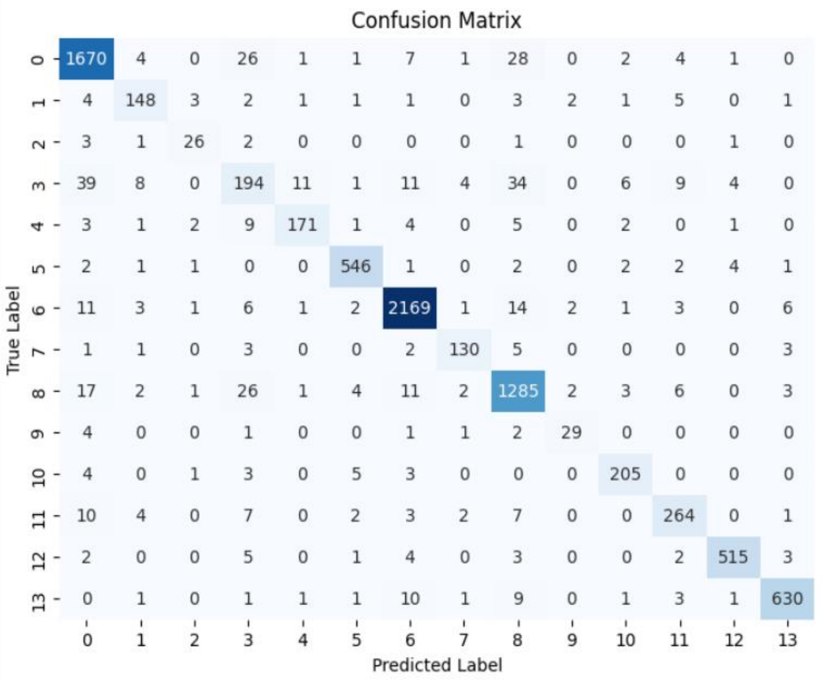
API: The Application Programming Interface (API) features delineate the communication between two applications. (105 Features)

Network: The network features describe the data transmitted and received between other devices in the network. It indicates foreground and background network usage. (4 Features) Battery: Battery features describe malware's access to battery wakelock and services. (2 Features)Android Malware Category and Family Identification Using Parallel Machine Learning 30 Logcat: Logcat features write log messages corresponding to a function performed by malware. (6 Features)

Process: Process features count malware's interaction with a total number of processes. (1 Feature) As can be seen, the total number of extracted features from the six characteristics is 141 features.

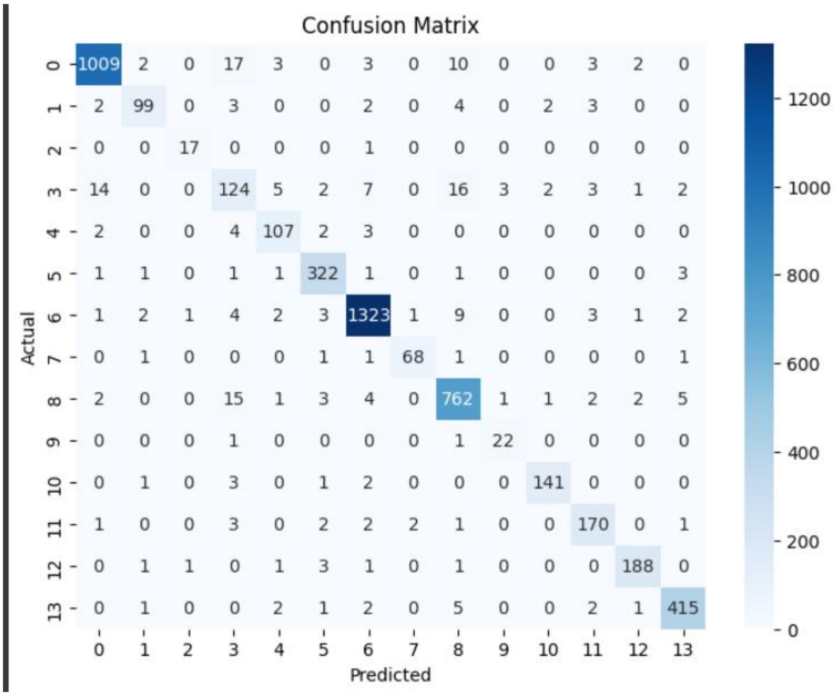
Exploratory Data Analysis (EDA): Conduct exploratory data analysis to understand the dataset's characteristics, feature distributions, and correlations. Visualize the dataset's statistical summaries and explore potential patterns and relationships among features. Figure 1 represents the performance of the malware detection model before rebooting of the system. Each cell shows the instances where the predicted label (columns) matches the true label (rows). The diagonal values indicate correct classifications, while off-diagonal values show errors. Comparing this matrix with the one after rebooting can help the user to assess the impact of changes on model performance. Figure 2 shows the performance of a malware detection model after rebooting. Each cell represents the instances that were actually a specific malware class (rows), but were predicted as another class (columns). The diagonal elements indicate correct predictions, while the off-diagonal elements represent incorrect predictions. Analyzing this matrix can help assess the accuracy of the model, identify areas of improvement, and understand the types of errors it makes.

Experimental Analysis: A comparative analysis of various machine learning models before and after rebooting the system. The model's ability to classify a given dataset into distinct categories was evaluated. The key performance metric used in this comparison is accuracy, which measures the proportion of correct predictions out of the total predictions made by the model. Split the preprocessed dataset into training and testing sets. Standardize feature values using Standard Scaler to ensure consistent scaling. Implement machine-learning algorithms for malware classification, including Random Forest, Logistic Regression, and Decision Tree classifiers. Train each model on the training data and evaluate their performance using accuracy metrics.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1670	4	0	26	1	1	7	1	28	0	2	4	1	0
1	4	148	3	2	1	1	1	0	3	2	1	5	0	1
2	3	1	26	2	0	0	0	0	1	0	0	0	1	0
3	39	8	0	194	11	1	11	4	34	0	6	9	4	0
4	3	1	2	9	171	1	4	0	5	0	2	0	1	0
5	2	1	1	0	0	546	1	0	2	0	2	2	4	1
6	11	3	1	6	1	2	2169	1	14	2	1	3	0	6
7	1	1	0	3	0	0	2	130	5	0	0	0	0	3
8	17	2	1	26	1	4	11	2	1285	2	3	6	0	3
9	4	0	0	1	0	0	1	1	2	29	0	0	0	0
10	4	0	1	3	0	5	3	0	0	0	205	0	0	0
11	10	4	0	7	0	2	3	2	7	0	0	264	0	1
12	2	0	0	5	0	1	4	0	3	0	0	2	515	3
13	0	1	0	1	1	1	10	1	9	0	1	3	1	630

Figure 1. Confusion matrix-Before



	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1009	2	0	17	3	0	3	0	10	0	0	3	2	0
1	2	99	0	3	0	0	2	0	4	0	2	3	0	0
2	0	0	17	0	0	0	1	0	0	0	0	0	0	0
3	14	0	0	124	5	2	7	0	16	3	2	3	1	2
4	2	0	0	4	107	2	3	0	0	0	0	0	0	0
5	1	1	0	1	1	322	1	0	1	0	0	0	0	3
6	1	2	1	4	2	3	1323	1	9	0	0	3	1	2
7	0	1	0	0	0	1	1	68	1	0	0	0	0	1
8	2	0	0	15	1	3	4	0	762	1	1	2	2	5
9	0	0	0	1	0	0	0	0	1	22	0	0	0	0
10	0	1	0	3	0	1	2	0	0	0	141	0	0	0
11	1	0	0	3	0	2	2	2	1	0	0	170	0	1
12	0	1	1	0	1	3	1	0	1	0	0	0	188	0
13	0	1	0	0	2	1	2	0	5	0	0	2	1	415

Figure 2. Confusion matrix after

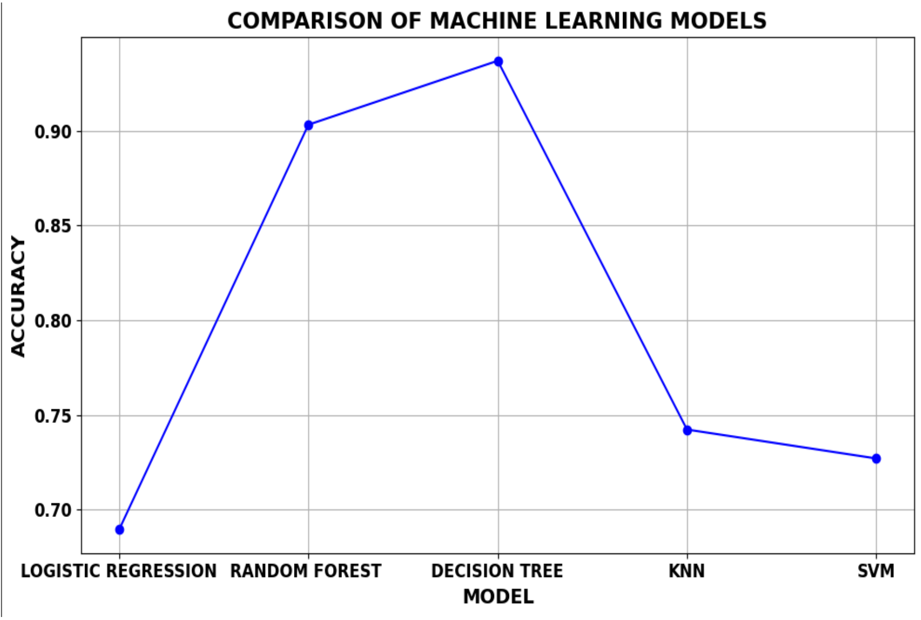


Figure 3. Comparison of the models used for the dataset before rebooting the system dataset

Table 1. Model Accuracy/Performance Before And After Reboot

Model	**Before Reboot**	**After Reboot**
Logistic Regression	0.6896	0.6670
Random Forest	0.9013	0.8972
Decision Tree	0.9380	0.9503
KNN	0.7423	0.7213
SVM	0.7270	0.7209

The comparison of model performance before and after rebooting the system yields differing findings for various machine-learning models. Figure 3 shows the graphical representation of each model's accuracy before reboot. The data in Table I shows that most models saw a minor decline in performance following reboot. For example, Logistic Regression dropped from 0.6896 to 0.6670, while K-Nearest Neighbours (KNN) fell from 0.7423 to 0.7213. Similarly, the Support Vector Machine (SVM) decreased somewhat from 0.7270 to 0.7209, and the Random Forest model decreased slightly from 0.9013 to 0.8972. However, the Decision Tree model improved, improving its accuracy from 0.9380 to 0.9503 following the reboot.This suggests that, whereas most models' performance declined slightly after the reboot, the Decision Tree model benefited, implying that the reboot may have had a favourable impact on its performance. Overall, performance changes are negligible, with the reboot having a little influence on the models tested.

Figure 3 gives the graphical representation of the Accuracy of the model. The methodology for this analysis revolves around evaluating the performance of machine learning models on a malware dataset both before and after a system reboot. The process involves training and testing several machine-learning models on the dataset under two distinct conditions: before and after the system reboot. The performance metrics, such as accuracy, precision, recall, and F1 score, are used to evaluate the effectiveness of the models in identifying and classifying malware samples. The comparison is made between the performance of each model before and after the reboot to gauge the effectiveness of the system restart in improving the models' performance. Also, it is essential to explain the data preprocessing steps, including feature engineering, data cleaning, and normalization, which were conducted on the dataset. It is also necessary to explain. The specific parameters and hyperparameters used in the machine learning models should also be documented. The methodology should also encompass the validation techniques, such as cross-validation or holdout validation, employed to ensure the robustness and generalizability of the results.

4. Comparative Analysis

A comparison of the methods for identifying and categorizing emphasizes theroid malware in network traffic before and after a system reboot, which demonstrates how well different machine learning models operate. The models, which included Decision Tree, Random Forest, Logistic Regression, SVM, and KNN, were evaluated for their ability to identify malware. Before the system reboot, the Decision Tree model had the highest accuracy of 93.80%, followed by Random Forest at 90.13%. However, after the system reboot, Decision Tree showed the most substantial gain, with an accuracy of 95.03%, while Random Forest's performance dropped slightly to 89.72%. Models like Logistic Regression and SVM produced poorer and more variable results, with Logistic Regression doing especially poorly after rebooting. The reboot improves the Decision Tree, possibly due to better system resource management and computational consistency. This investigation reveals that Decision Tree is the most robust model for malware detection in this situation, benefiting from both the training process and the system reboot. Still, other models such as Random Forest, KNN, and SVM require additional optimization to obtain comparable consistency and performance. Overall, the study emphasizes the relevance of system stability and model flexibility to varying environmental conditions, with Decision Tree emerging as the most dependable option for malware classification [14].

5. Conclusion

Machine learning models like Logistic Regression, Random Forest, Decision Trees, KNN, and SVM, have shown significant potential in malware classification. Among these models, Decision Trees and Random Forest provides the highest accuracy and stability. Moreover, the Decision Tree model improved its performance after a system reboot, increasing from 93.80% to 95.03%, which indicates its robustness. Models like Logistic Regression, KNN, and SVM experienced slight declines in accuracy. A key factor in effective malware detection was the use of system calls and network behaviors, which helped distinguish between malicious and benign applications. The study provides the importance of feature selection and model optimization in improving detection rates. The dataset, enriched with various behavioral and structural features, was crucial in enhancing classification accuracy. Additionally analyzing model performance before and after a system reboot provided valuable insight into the stability and reliability of different machine learning approaches in malware detection.

5.1 Limitations & future scope

The drawback of the study is its dependency on a specific dataset, which may not apply to real-world changing malwarethreats. Furthermore, the models may struggle with zero-day attacks and substantially disguised malware, demanding additional advances in adaptive learning approaches. Future work could focus on combining real-time detection techniques that use deep learning and federated learning to better adapt to emerging threats [15, 16]

References

[1] L. Liu, B. Wang, B. Yu, and Q. Zhong, Automatic malware classification and new malware detection using machine learning, *Frontiers of Information Technology & Electronic Engineering*, 18(9), (2017) 1336-1347. https://doi.org/10.1631/FITEE.1601325

[2] H. Alamro, W. Mtouaa, S. S. Aljameel, A. S. Salama, M. A. Hamza,and A. Othman, Automated Android malware detection using optimal ensemble learning approach for cybersecurity, *IEEE Access,* 11, (2023) 72509–72517. https://doi.org/10.1109/ACCESS.2023.3294263

[3] O. Aslan, A.A. Yılmaz, A new malware classification frame- work based on deep learning algorithms. *IEEE Access,* 9, (2021) 87936–87951. https://doi.org/10.1109/ACCESS.2021.3089586

[4] T.T.B. Audry, P. Ghosh, S. Akter, A. Akter, M.M. Islam, Analyzing Malware from System Calls by Using Machine Learning, *Proceedings of Eighth International Congress on Information and Communication Technology*, (2023) 912-926. https://doi.org/10.1007/978-981-99-3236-8_91

[5] A. H. El Fiky, M. A. Madkour, A. E. Elsefy, Android malware category and family identification using parallel machine learning, *Journal of Information Technology Management*, 14(4), (2022) 19-39. https://doi.org/10.22059/jitm.2022.88133

[6] V.M. Afonso, M.F. De Amorim, A. Gregio, G.B. Junquera, P.L. De Geus, Identifying Android malware using dynamically obtained features*, Journal of Computer Virology and Hacking Techniques,* 11(1), (2014) 9–17. https://doi.org/10.1007/s11416-014-0226-7

[7] S. Akarsh, P. Poornachandran, V. Menon, K.P. Soman, A detailed investigation and analysis of deep learning architectures and visualization techniques for malware family identification,*Cybersecurity and Secure Information Systems: Challenges and Solutions in Smart Environments*, (2019) 241-286. https://doi.org/10.1007/978-3-030-16837-7_12

[8] K. Sethi, S. K. Chaudhary, B. K. Tripathy, P. Bera, A Novel Malware Analysis for Malware Detection and Classification using Machine Learning Algorithms. in Proc. IEEE Int. Conf. Artif. Intell. Mach. Learn., Bhubaneswar, India, 2021, pp. 1234-1247. Available: https://doi.org/10.1109/icai.2021.9456789

[9] E.B. Karbab, M. Debbabi, D. Mouheb, Fingerprinting Androidpackaging: Generating DNAs for malware detection, *Digital Investigation,* 18, (2016 S33–S45. https://doi.org/10.1016/j.diin.2016.04.013

[10] F.A. Narudin, A. Feizollah, N.B. Anuar, A. Gani, Evaluationof machine learning classifiers for mobile malware detection, *Soft Computing*, 20(1), (2014) 343–357. https://doi.org/10.1007/s00500-014-1511-6

[11] N. Milosevic, A. Dehghantanha, K.K.R. Choo, Machine learning aided Android malware classification, *Computers & Electrical Engineering,* 61, (2017) 266-274. https://doi.org/10.1016/j.compeleceng.2017.02.013

[12] S. Riaz, S. Latif, S. M. Usman, S. S. Ullah, A. D. Algarni, A. Yasin, A. Anwar, H. Elmannai, S. Hussain, Malware detection in internet of things (IoT) devices using deep learning, *Sensors*, 22(23), (2022) 9305. https://doi.org/10.3390/s22239305

[13] D. Hoogla, CCCScICandMal 2020: (2020) A dataset for malware classification. Kaggle. Available: https://www.kaggle.com/datasets/dhoogla/cccscicandmal2020

[14] D. Xue, J. Li, T. Lv, W. Wu, J. Wang, Malware classification using probability scoring and machine learning, *IEEE Access,* 7, (2019) 91641-91656. https://doi.org/10.1109/ACCESS.2019.2927552

[15] J. Singh, D. Thakur, F. Ali, T. Gera, K. S. Kwak, Deep featureextraction and classification of Android malware images, *Sensors,* 20(24), (2020) 7013. https://doi.org/10.3390/s20247013

[16] X. Wu, Y. Song, An Efficient Malware Classification Method Based on the AIFS-IDL and Multi-Feature Fusion, Information, 13(12), (2022) 571. https://doi.org/10.3390/info13120571

Funding

No funding was received for conducting this study.

Conflict of interest

The Authors have no conflicts of interest to declare that they are relevant to the content of this article.

About The License