Detecting Malware Families and Subfamilies using Machine Learning Algorithms: An Empirical Study

Esraa Odat and Batool Alazzam
Faculty of Computer and Information Technology
Jordan University of Science and Technology
Irbid, Jordan 22110

Qussai M. Yaseen*
College of Engineering and Information Technology,
Ajman University, UAE
Faculty of Computer and Information Technology
Jordan University of Science and Technology, Irbid, Jordan 22110

***Abstract*—Machine learning algorithms have proved their effectiveness in detecting malware. This paper conducts an empirical study to demonstrate the effectiveness of selected machine learning algorithms in detecting and classifying Android malware using permissions features. The used dataset consists of 9000 different malicious applications from the CIC-Maldroid2020, CIC-Maldroid2017 and CIC-InvesAndMal2019 datasets collected by the Canadian Institute for Cybersecurity. Meta-Multiclass and Random Forest ensemble classifiers are used based on different machine learning classifiers to overcome the imbalance in the data classes. Moreover, a genetic attribute selection technique and SMOTE are used to classify Ransomware sub-families to handle the small size of the dataset and underfitting problem. The results show that optimization and ensemble approaches are successful in treating dataset issues, with 95% accuracy in classifying big malware families and 80% in Ransomware subfamilies.**

***Keywords*—Malware classification; machine learning; SMOT; information security**

I. Introduction

Malware is a malicious software that aims at affects the confidentiality, integrity or availability of data and systems without users consent to attain the harmful intent of the attacker [1] [2]. Malware applications are classified into many classes according to their behaviour and properties such as adware, worms, viruses, rootkits, trojan horse, backdoor, spyware, logic bombs, adware, and ransomware. Systems resources are attacked to affect the assets for the purposes of getting financial benefits, for stealing private information or using the computing resources to attack other victims [3] [4] [5].

The usage of smartphone devices are growing immensely, which provides attackers a powerful mean to access users private information. According to google [6], there were 2 billion Android devices until November 2017, which means that Android operating system has 71.15% of the Mobile Operating System Market Share Worldwide [7] [8]. The wide spread of Android devices has increased Android attacks three times for the past two years. Therefore, there is a significant need to find ways to detect and classify malware families.

Fig. 1 shows the mobile malicious installation packages for Android that were discovered by Kaspersky Company only between 2017 to 2020, which shows an increase in the discovered malware from 2019 to 2020 by more than 2,100,000 malicious packages [9].

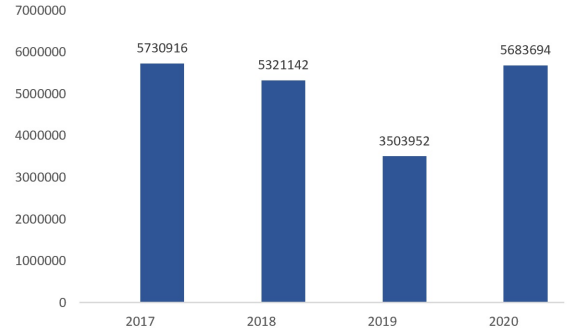

Fig. 1. Mobile Malicious Installation Packages for Android in 2017 through 2020

Android malware apps can be classified using static analysis method, dynamic analysis method, and Hybrid approach. In Static analysis method, the static features such as static APIs and permissions are used to classify APKs into malware or benign applications. Meanwhile, dynamic analysis approach uses dynamic features such as dynamic APIs, memory usage, CPU usage,Network outgoing traffic, etc. to classify malware and benign applications. Extracting dynamic features of malware is performed using an isolated environment, called sandbox, such as cuckoo sandbox [10]. Fig. 2 shows the environment of dynamic analysis of android applicatoins and extracting dynamic features. The hybrid approach uses a combination of static features and dynamic features to classify Android APKs.

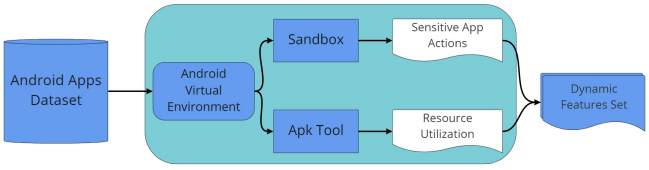

Fig. 2. Dynamic Analysis of Android Apps

The aforementioned approaches may be used with hyperparameter tuning to perform correctly. Similarly, Other approaches are used to deal with the underfitting problem, such as genetic algorithms [11]. The Genetic Algorithm (GA) is a search-driven optimization technique based on genetic and natural selection principles. It is often used to find almost the optimal solutions for complex problems [6]. Furthermore,

Malware analysis using machine learning may face dataset oversampling problems. For example, synthetic Minority Oversampling Technique (SMOTE) [12], which is an oversampling technique, is used to generate samples for the minority class. It cares about the feature space to generate new samples and helps of interpolation between positive samples that belong to each other. SMOTE selects the first positive class randomly then KNN's for the chosen positive sample is gained [7].

This paper implements and compare the performance of several machine learning algorithms in classifying a large dataset of APKs into malware and benign applications, classifying malware applications into its main classes, and classifying ransomware applications into its subfamilies. The used datasets are CIC-Maldroid2020, CIC-Maldroid2017 and CIC-InvesAndMal2019, which consists of 9000 Android APK samples were collected by the Canadian Institute for Cybersecurity.The measures used are f-score, accuracy, TPR and FPR. The contributions of this paper are summarized as follows:

- It classifies the dataset into malware or benign application using various machine learning algorithms.
- It classifies the malware into main classes: SMS malware, Ransomware, Banking malware, Scareware, Adware, and Premium SMS malware.
- It classifies the Ransomware family as subfamilies depending on extracted permissions as static features.
- It employs optimization techniques to overcome the problems dataset: ensemble algorithms, SMOTE, and genetic algorithm. The ensemble machine learning algorithms. The meta-Multiclass and Random Forest classifiers are used to solve unbalancing problems in classifying malware into its main classes to get accurate and unbiased performance measures. Furthermore,the paper applies the genetic algorithm and SMOTE used to improve the performance of classifying ransomware subfamilies because it is a small and imbalance dataset.

The rest of paper is organized as follows. The next section discusses some related work. Section 3 demonstrates the used methods and materials. Section 4 demonstrates and discusses the results. Finally, Section 5 concludes the work.

II. Related Work

The use of machine learning algorithms in detecting and classifying malware have been studied widely. However, due to the continuous growing of malware types, the change in their features, and the skills of attackers, machine learning algorithms and the features they use need to be optimized an modified continuously. This section discusses some related work in this field.

Zhiwu et al. [13] proposed a classification method based on static features, which are bytecode features, assembler code features, and PE features. The authors used eight machine learning classifier models, which are Gaussian Naïve Bayes, Random Forest, Decision Tree, SVM linear, SVM, KNN, and Ada-boosting. They used a dataset from VirusShare. In the comparison among the different algorithms, the cost, the needs, and convenience were considered. In addition, they used Kaspersky scan engine results. According to the experiments, the Random Forest model achieved the highest result of the F1 score of 93.56% .

Fang et al. [14] proposed an approach based on extracting control-flow graphs (CFG) and data-flow graphs (DFG) during the instruction level. Then, they encoded the graphs into matrices and used them to build the family classification model using deep learning. The family classification model considered the horizontal combination of CFG and DFG as features to achieve the best performance. The malware dataset used in the experiments were collected from Marvin, Drebin, VirusShare and Contagio-Dump. The results showed that the horizontal combination of CFG and DFG performed better than CDGDroid.

Arslan et al. [15] proposed a method for Android malware families classification using Dalvik Executable (DEX) file section features. The proposed approach converted DEX files to Red/Green/Blue images and plain text. Next, from these images, texture of the image, the color, and text were extracted as features. The results showed that the proposed method achieved a precision of 96%.

Gandotra et al. [16] used the manifest file to extract the permissions as features of android applications. The source code was used to verify whether a permissions is requested by the application or not. The considered the assumption that malicious apps are those that request several different permissions without using them. The proposed approach extracted all permissions requested by an application and stored them in a database. Then, the use of permissions was verified. Next, for each app, the proposed approach computed the number of suspicious values. These values were used for classification. The results showed that the SVs for malicious apps and benign apps were in the ranges [4-95,858] and [21-55,967] respectively. To classify the apps, the approach used Naive Bayes and Logistic Regression. The authors claimed that the accuracy was 91.95%.

Şahın et al. [17] proposed an approach to detect zero-day malware based on the integration of static and dynamic analysis. The authors validated the proposed approach using a real-world dataset of malicious samples. The performance of the proposed approach was measured before and after features selection to show the effectiveness of the proposed relevant feature selection method in improving the model construction time as well as the accuracy. The proposed approach used the entropy to compute the purity of each feature and select the relevant features. The results showed that the accuracy of all classifiers using the integrated features set is very good. However, the naive Bayes classifier did not achieve good results. Furthermore, the results showed that the approach of features selection improved the model building time and the accuracy.

Udayakumar [18] suggested a new permission-weight approach. The approach applied the algorithms of KNN and Naive Bays to build the model and used RF as a weighting matrix to assign weights for permissions based on whether a permission is requested by the app (benign and malicious apps). The results showed that by adding the weighted approach, the KNN algorithm improved 2% and the NB algorithm improved 7%.

Milosevic et al. [19] proposed an approach that help in understand the types of malware, and machine learning algorithms, such as Decision Trees, Multilayer Perceptron and Multi SVM, can be used to detect malware. The proposed approach used the debug Size as a feature with a score of 0.26. Using SVM and neural networks for classification, the proposed approach got an of 90.2% and 98% at training approach, and neural networks get 99% at the testing approach.

Alzubaidi [20] applied an approach based on deep learning algorithms to detect android malware apps families. He claimed that he achieved an accuracy of 99%. Meanwhile, Ashit [21] applied an enhanced Birch algorithm to find the malware and modified executables of Windows and Android operating system.

III. Methods and Materials

The method proposed in this paper is divided into two parts: classifying main malware-families and classifying Ransomware subfamilies. The classification process is based on using permissions as features to show the ability of machine learning classifiers to distinguish between the different malware families. Furthermore, it aims at helping security officers to develop more reliable systems against different types of malware, and to detect malicious applications by using only the permissions and minimum resources.

A. Features Extraction

The "apktool.jar", which is a reverse engineering tool, was used to extract the permissions of each APK. As a result, 140 permissions were extracted. Next, a python code was developed and used to represent the APKs and the requested permissions by each APK in a vector space file.

Fig. 3 shows the operations of the extracting and selecting permissions features. The figure shows that the APK tool was used to analyze the Android APK file in order to extract the Manifest.xml and classes.dex files. Next, the Manifest parser was used to extract the permission features, from which the used features were selected form the classification process.

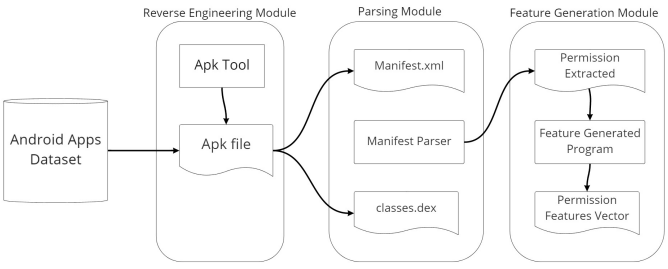

Fig. 3. Extracting and Selecting Permissions Features

B. Dataset

Two datasets have been used in this work. The first dataset is CIC-Maldroid2020 [23] collected from Canadian Institute for Cybersecurity. The dataset consists of 8750 Android samples as APK files, while the second dataset consists of 250 ransomware APKs collected from CIC-InvesAndMal2019 [22] and CIC-Maldroid2017 [24]. The dataset was divided into three subsets. The first dataset consists of 1422 Adware, 2506 Banking, and 4821 SMS malware. The second dataset consists of 250 Ransomware subfamilies.

C. Methodology

The work has two main parts. The first one is the feature extraction of APKs permissions as well as feature selection. The second one is the data processing, where main malware families were classified using ensemble machine learning algorithm; Random Forest, Decision Tree and meta-Multiclass classifiers. Meanwhile, Naïve Bayes, KNN, Decision Tree, and Logistic Regression were used to classify Ransomware family as sub-families.

IV. Results and Analysis

A. Main Malware Family Classification

The classification of malware APKs into main classes were performed using Meta-Multiclass, which is a classifier for handling multi-class datasets with 2-class classifiers. Meta-Multiclass has various capabilities including error correcting output codes to increase the accuracy. Using this method, when the base classifier cannot handle instance weights because they are not uniform, the data is re-sampled with replacement based on the weights before being processed by the base classifier.

Table I shows the performance matrices for the main malware families classification. The result shows that the Meta-Multiclass classifier achieved high performance by classifying malware with 94% accuracy and low error rate of about 0.029.

Fig. 4 shows classification error for meta-Multiclass classifier using KNN as base classifier. The x-axis and y-axis represent the main malware family names, where 1, 2 and 3 labels denotes Adware, SMS and Banking families respectively, star shapes represent correct predicted points and square shapes represent incorrect predicted points. The results show that the highest prediction power of the classifier is for Adware data samples with 97% true positive rate.

TABLE I. Meta-Multiclass Classifier

Malware Family	TPR	FPR	F -SCORE	ROC
Adware	0.968	0.018	0.966	0.988
SMS	0.872	0.022	0.910	0.954
Banking	0.985	0.047	0.947	0.979
AVG	0.942	0.029	0.941	0.974

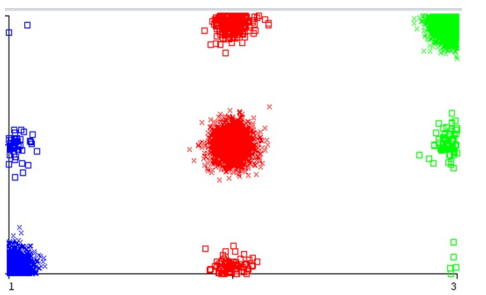

Fig. 4. Classification Error of Meta-Multiclass Classifier using KNN base Classifier

Fig. 5 shows the classification error for Random Forest classifier. The x-axis and y-axis represent main malware family names, where 1, 2 and 3 labels denotes Adware, SMS and Banking families respectively, star shapes represent correct predicted points and square shapes represent incorrect predicted points. To proof the meta-Multiclass findings, random forest ensemble classifier on decision tree as a base classifier was applied. According to the results, the two used ensemble methods showed similar performance.

Using Meta-Multiclass classifier based on KNN as base classifier and Random Forest classifier, the accuracy of ensemble models are 95%. Furthermore, the results show that the ensemble classifiers handle the data unbalancing problem and classify main malware families correctly. The values of TPR, F-score, accuracy, and ROC show the high performance and low FPR value of about 0.027 as shown in Table II.

TABLE II. META-MULTICLASS CLASSIFIER BASED ON KNN AS BASE CLASSIFIER AND RANDOM FOREST CLASSIFIER

Malware Family	TPR	FPR	F -SCORE	ROC
Adware	0.977	0.018	0.970	0.993
SMS	0.875	0.019	0.915	0.964
Banking	0.983	0.045	0.949	0.981
AVG	0.945	0.027	0.945	0.979

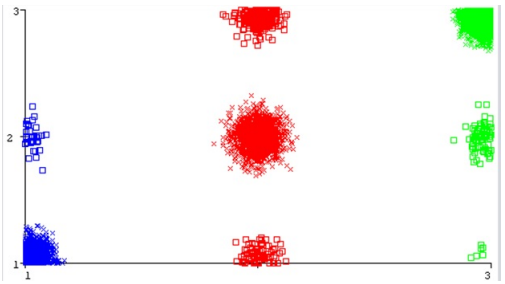

Fig. 5. Classification Error of Random Forest Classifier

B. Ransomware Subfamilies Classification

The dataset used in this experiment was imbalanced. Therefore, the challenge was to apply Machine learning algorithms to measure the effectiveness of ML different models in distinguishing ransomware subfamilies and treating the data problems.

1) Ransomware Subfamilies Classification using Original Dataset: To evaluate the effectiveness of machine learning algorithms in detecting Ransomware subfamilies, the Decision tree classifier was applied on the original dataset.

Fig. 6 shows the classification error for Decision Tree classifier on original dataset, were the x-axis and y-axis represent Ransomware subfamily names, star shapes represent correct predicted points and square shapes represent incorrect predicted points.

Table III shows that using Decision Tree classifier, the accuracy are 62%. That is, the results show that machine learning algorithms based on permissions requested by applications on original dataset does not effectively classify Ransomware subfamilies.

TABLE III. DECISION TREE CLASSIFIER

Sub-family	TPR	FPR	F-score	ROC
Wannalocker	0	0.02	0	0.88
Svpeng	0.8	0.13	0.6	0.89
Simplocker	0.5	0.03	0.52	0.79
RansomBO	0.6	0.06	0.5	0.94
PornDroid	0.6	0.13	0.55	0.87
Pletor	0.5	0.04	0.5	0.88
LockerPin	0.8	0	0.9	0.91
Koler	0.5	0	0.66	0.91
Jisut	0.9	0.01	0.92	0.98
Charger	0.8	0.01	0.84	0.99
Average	0.6	0.06	0.61	0.9

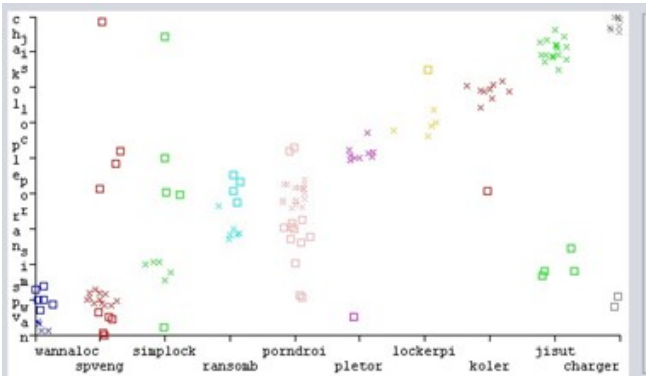

Fig. 6. Classification Error of Decision Tree on Original Data

2) Ransomware Subfamilies Classification using Optimization Algorithms: To address data underfitting and imbalance issues, we used Synthetic Minority Oversampling Technique (SMOT), which uses the KNN algorithm to generate new observations to eliminate the imbalance. In addition, we used the genetic algorithm as hyperparameter tuning or parameter tuning. Furthermore, the paper used the ensemble learning algorithm to increase the model chance learning and produce a good prediction.

Fig. 7 shows the classification error for Random Forest classifier with optimization algorithms, where x-axis and y-axis are Ransomware subfamily names, star shapes represent correct predicted points and square shapes represent incorrect predicted points. The imbalance and underfitting problems were almost solved after applying SMOT and the genetic algorithms with the random forest classifier. We noticed that the accuracy improved up to 80% from the previous results, which was 60%, with low false positive rate with 0.028 as shown in Table IV.

TABLE IV. RANDOM FOREST CLASSIFIER

Sub-family	TPR	FPR	F-score	ROC
Wannalocker	0.4	0.05	0.39	0.94
Svpeng	0.7	0.05	0.69	0.97
Simplocker	0.9	0	0.94	0.96
RansomBO	0.7	0.07	0.54	0.95
PornDroid	0.7	0.03	0.78	0.97
Pletor	1	0	1	1
LockerPin	0.8	0	0.9	0.99
Koler	0.9	0	0.94	0.99
Jisut	1	0.02	0.95	0.99
Charger	1	0	1	1
Average	0.8	0.03	0.8	0.97

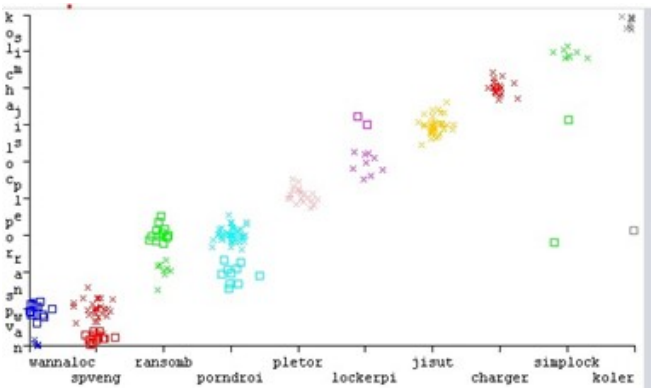

Fig. 7. Classification Error of Random Forest with Optimization Algorithms

V. Conclusion

Android threats and attacks are rapidly increasing as Android devices and the number of users increasing around the world. Therefore, the attacks on android operating systems and users private information has increased. This paper has discussed this issue and used an optimized machine learning approach based on permissions as features to detect malware. The paper has used a large dataset to detect malware and classify detected malware into main malware families, which are SMS malware, Banking malware, Adware. In addition, the optimized approach was tuned to classify ransomware dataset into its subfamilies. The optimized approach used ensemble classifiers such as meta-Multiclass classifier with KNN as base classifier and Random forest to classify main malware families and handling the unbalanced dataset. In addition, the optimized approach used Random Forest and Decision Tree classifiers to classify ransomware subfamilies. The results have shown that ensemble classifiers perform very well in handling unbalanced data, detecting and classifying main malware families by achieving an accuracy of 95%. Furthermore, the results have shown that detecting and classifying Ransomware subfamilies using traditional machine learning algorithms has a poor performance with an accuracy of 62% for Decision Tree. However, the tuned approach by the oversampling technique has increased the accuracy to 81%.

References

[1] C. K. A. Moser en E. Kirda, “Limits of Static Analysis for Malware Detection”, in 23rd Annual Computer Security Applications Conference, 2007, pp 421–430.

[2] E. Kirda, “Dynamic Analysis of Malicious Code”, Journal in Computer Virology, vol 2, pp 67–77, 2006.

[3] “A comparison of static, dynamic, and hybrid analysis for malware detection”, Journal in Computer Virology, vol 13, pp 1–24, 2017.

[4] E. M. Dovom, A. Azmoodeh, A. Dehghantanha, D. E. Newton, R. M. Parizi, en H. Karimipour, “Fuzzy pattern tree for edge malware detection and categorization in iot”, Journal Systems Architecture, vol 97, pp 1–7, 2019.

[5] M. Ficco en F. Palmieri, “Leaf: An open-source cybersecurity training platform for realistic edge-iot scenarios”, Journal Systems Architecture, vol 97, pp 107—129, 2019.

[6] B. Popper, “https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users”, 2017 (accessed,2021).

[7] StatCounter, “Mobile Operating System Market Share Worldwide”, 2020 (accessed,2021).

[8] B. Sun, Q. Li, Y. Guo, Q. Wen, X. Lin, en W. Liu, “Malware family classification method based on static feature extraction”, in 2017 3rd IEEE International Conference on Computer and Communications (ICCC), 2017, pp 507–513.

[9] Kaspersky, “Malware Growth”, Available: https://securelist.com/mobile-malware-evolution-2020/101029/. (accessed,2021).

[10] “Cuckoo Sandbox”.Available: https://cuckoosandbox.org,(accessed,2021).

[11] Rodkaew Y., “A Genetic Algorithm as a Classifier”, in 18th International Conference on Electrical Engineering/Electronics , Computer, Telecommunications and Information Technology (ECTI-CON), 2021, pp 254–257.

[12] “SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary”, Journal of Artificial Intelligence Research, vol 61, pp 863–905, 2018.

[13] X. U. Zhiwu, K. Ren, en F. Song, “Android malware family classification and characterization using CFG and DFG”, in 2019 International Symposium on Theoretical Aspects of Software Engineering (TASE), 2019, pp 49–56.

[14] Y. Fang, Y. Gao, F. Jing, en L. Zhang, “Android malware familial classification based on DEX file section features”, IEEE Access, vol 8, pp 10614–10627, 2020.

[15] R. S. Arslan, İ. A. Doğru, en N. Barişçi, “Permission-based malware detection system for android using machine learning techniques”, International journal of software engineering and knowledge engineering, vol 29, no 01, pp 43–61, 2019.

[16] E. Gandotra, D. Bansal, en S. Sofat, “Zero-day malware detection”, in 2016 Sixth international symposium on embedded computing and system design (ISED), 2016, pp 171–175.

[17] D. Ö. Şahın, O. E. Kural, S. Akleylek, en E. Kiliç, “New results on permission based static analysis for Android malware”, in 2018 6th International Symposium on Digital Forensic and Security (ISDFS), 2018, pp 1–4.

[18] N. Udayakumar, V. J. Saglani, A. V. Cupta, en T. Subbulakshmi, “Malware classification using machine learning algorithms”, in 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), 2018, pp 1–9.

[19] N. Milosevic, A. Dehghantanha, en K.-K. R. Choo, “Machine learning aided Android malware classification”, Computers and Electrical Engineering, vol 61, pp 266–274, 2017.

[20] A. Alzubaidi, “Sustainable Android Malware Detection Scheme using Deep Learning Algorithm”, International Journal of Advanced Computer Science and Applications, vol 12, pp 860–867, 2021.

[21] A. Dutta, “Detection of Malware and Malicious Executables Using E-Birch Algorithm”, International Journal of Advanced Computer Science and Applications, vol 7, pp 124—126,

[22] Canadian Institute of Cybersecurity, “CIC-InvesAndMal2019”, Available: https://www.unb.ca/cic/datasets/invesandmal2019.html, 2019 (accessed,2022).

[23] Canadian Institute of Cybersecurity, “CIC-Maldroid2020”, Available: https://www.unb.ca/cic/datasets/maldroid-2020.html, 2020 (accessed,2022).

[24] Canadian Institute of Cybersecurity, “CIC-Maldroid2017”, Available: https://www.unb.ca/cic/datasets/andmal2017.html, 2017 (accessed, 2022)