Android Malware Detection and Familial Classification using Dynamic Features for Imbalanced Dataset

Swapna Augustine Nikale[1], Dr. Seema Purohit [2]
[1] Research Scholar, Department of Computer Science, University of Mumbai, Maharashtra, India.
[2] Research Guide, Department of Computer Science, University of Mumbai, Maharashtra, Inida.
Email: [1] swapna.nikale@gmail.com, [2] supurohit@gmail.com

Abstract
Mobile phones are one of the widely used communication platforms. It is also a much-preferred device to carry out various diverse activities in different sectors such as gaming, education, finance, the stock market, etc. Android operating system is one the most popularly used mobile operating system which requires mobile applications to perform any activity. Due to its open-source nature and large consumer market share, many illegitimate operations are targeted specifically through malware mobile applications. The objective of this paper is to obtain resilient features from Android APK files, to analyze the effectiveness of various machine learning classifiers, to perform hyperparameter tuning to find the appropriate parameters of the classifiers that improve the evaluation metrics, and to handle the imbalance issue in the dataset using various imbalance approaches.
Index Terms: android malware detection, class imbalance, dynamic features, machine learning classifiers.

1. Introduction

Android operating system, owned by Google, is one of the widely used and preferred operating systems among mobile phone users. Due to its ease of use and open-source nature, Android is also used in various devices ranging from TVs to smart appliances. Android dominates the worldwide market share with 71% usage which makes it on the much-preferred list of operating systems [1].

Many users prefer mobile phones rather than laptops and desktops to proceed with their day-to-day activities such as gaming, banking, stock trading, attending lectures, bill payments, etc. From 2010 till 2023, every year there is at least a 25% increase in mobile phone usage among consumers [2].

Android attracts the largest consumers due to its user-friendly characteristics. It has also become an attraction to many attackers to outbreak the open-source platform of Android. The first Android malware named DroidSMS was released in 2010 and caused havoc by subscribing users to premium SMS without authorization [3]. Since then, the rise of malware on mobile phones see a steady rise that causes disruption to day-to-day activities and

compromises the privacy of mobile phone users. In the first quarter of the year 2023, we have already identified around 11 lakhs of new malware [4], [5].

Many security solutions were developed and introduced in the market by official partner Google and by third parties such as Bitdefender, Norton, Avast, Kaspersky, etc [6]. Google provides security solutions like Play Protect and Bouncer which routinely scans the mobile applications available in the official market for the presence of any malicious applications. Malware authors have bypassed such security mechanisms with various stealthy techniques [7], [8]. Malware authors take advantage of third-party play store that lacks such security solutions and uploads their malicious applications. They attract mobile phone users to download the mobile application from untrusted stores and infect the mobile device. Third parties provide a wide range of security solutions such as antivirus, antispyware, honeypot, etc. But these solutions operate and scan based on a traditional signature-based mechanism which could be easily bypassed by current stealthy malware.

Thus, the need for android malware detection based on a behavioristic approach is much in need to capture stealthy malware. This paper presents the process of obtaining the dynamic malware characteristics from mobile applications and demonstrates how it can be used to train a machine learning classifier to identify malicious android malware applications. It also highlights the importance of analyzing the characteristic of class imbalance in the dataset and how it can improve the accuracy of the machine learning classifier when handled correctly. The research idea proposed in the paper also identifies the mobile applications based on five families, adware, ransomware, smsware, scareware, and benign.

The paper is further divided into the following sections. Section II Study of Existing Approach Section III Proposed Research Methodology Section IV Experiments and Results.

2. Study of Existing Approach

Recent malware bypasses traditional security solutions through advanced approaches like dynamic loading and obfuscation techniques. The traditional security solutions that operate on signature-based techniques fail to identify such dynamically changing malware behavior [9].

The study presented in [10] performed android malware detection using Support Vector Machine (SVM) classifier using dynamic features and applying a dimensionality reduction approach to ease the complexity of large dimensions in the dataset. The study reported 89% accuracy in android malware detection.

The authors in [11] presented a static analysis-based android malware detection using Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) classifiers for Android malware detection and attained 83% accuracy.

In [12], the authors collected binder calls as dynamic features and applied various machine learning classifiers, and reported an accuracy of 86.36%.

The research study in [13] collected permissions and API calls and trained Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), and Naïve Bayes (NB) and developed a binary classification of Android malware detection.

The authors in [14] presented an Android malware classification approach with the help of static and dynamic features. The study reported on the comparative analysis of machine learning classifiers and neural networks.

In [15], the authors collected operating system-based features such as memory dump, and CPU usage and developed a binary classification of Android malware.

The study in [16] presented a binary classification of Android malware detection using API calls and system call traces using tree-based machine learning classifiers such as Decision Tree, Random Forest, etc.

The authors in [17] presented a comparative study on the performance of Android malware detection on real mobile devices and emulators. Around 24% of applications were correctly identified on mobile phones. The detection approach is binary classification.

The study in [18] stated that Android malware detection is based on static and dynamic features and trained various machine learning classifiers.

The research gap found based on the existing study was that all the detection approaches use only a specific machine learning classifier, and an elaborative experimental study on various machine learning classifiers was missing. Most of the research concentrated on binary classification rather than familial classification. The study of the imbalance dataset was missing which plays a major role in improving the accuracy of the detection and classification model. The dataset used is either obsolete or easily available which is a major drawback as most of the detection mechanisms are developed based on these data and recent malware is rewritten that bypass such security mechanisms.

The proposed methodology in our research paper meets the above shortcomings and develops a resilient solution for Android malware detection and familial classification.

3. Research Methodology

The research methodology presented in this paper is categorized into four phases. The first phase specifies the collection of a dataset and categorizing it into five families such as adware, ransomware, smsware, scareware, and benign, the second stage specifies the procedure for extracting the dynamic features from the dataset collected in the first phase, the third phase includes the process of training, testing and analyzing the results of various machine learning classifiers and the final fourth phase includes the process of tuning the hyperparameters to improve the accuracy of machine learning classifiers. Fig 3.1 presents the proposed research methodology.

A. Collection of Dataset

Android APKs were collected from various third-party sources such as Contagio Mobile [19], Koodous [20], and APKpure [21], for malware samples and from the Google Play store [22] for benign samples. The APKs were recent and alive since 2018 which played a major role in obtaining the recent stealthy malware behavior from Android APK files. The APKs were scanned using Virustotal which is one of the popular and widely used online scanners among security researchers that scanned APKs and provided the report of around 70 antivirus scanner services. Research conducted in [23] stated that the results provided by Virustotal were trustworthy and much accurate. Based on the report provided by Virustotal, the APKs

were categorized into five classes such as adware, ransomware, smsware, scareware, and benign

B. Extraction of Dynamic Features

The dataset contained 525 APK samples of which adware comprised 95 samples, ransomware 93 samples, smsware 108 samples, scareware 111 samples, and benign 118 samples. The dynamic features such as system calls and binder calls were extracted using the Mobile Security Framework (MobSF) tool [24]. This tool performed the dynamic analysis of the APK with an emulator and generated instances which captured the syscall and binder call. Totally 470 features of syscalls and binder calls were extracted. The frequency of each system calls and binder call was calculated and recorded in a CSV file for further analysis.

C. Training and Testing Machine Learning Classifiers

The extracted frequency of syscalls and binder calls were used to train and test the android malware detection and familial classification using machine learning models such as Logistic Regression, Decision Tree, XGBoost, K - Nearest Neighbor, Naïve Bayes, Support Vector Machine, and Random Forest.

D. Tuning hyperparameters to improve the accuracy

The results were recorded for each of the machine learning models tried and analyzed. Random Forest and XGBoost were able to provide better results compared to other machine learning models. The Random Forest and XGBoost further experimented with different hyperparameters. The effect of class imbalance on the dataset was studied and analyzed using SMOTE, ADASYN, and Balanced Cost approaches.

4. Experiments and Results

Experiments were conducted in the Google Colab environment using various python libraries. Scikit-Learn, Tensorflow, and Torch were used for the implementation of the machine learning classifier, and Matplotlib and Seaborn were used for visualization.

Classifiers like Logistic Regression, Decision Tree, XGBoost, K - Nearest Neighbor, Naïve Bayes, Support Vector Machine, and Random Forest were experimented with. Results were analyzed using evaluation metrics such as accuracy, precision, recall, and f1-score.

A brief description of the evaluation metrics is given below. TP (True Positive) represents a malware APK correctly classified as malware, TN (True Negative) represents a benign APK correctly classified as benign APK, FP (False Positive) represents a benign APK incorrectly classified as malware APK, and FN (False Negative) represents malware APK incorrectly classified as benign APK.

Accuracy presents the ratio of the total number of correctly classified APKs.

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision presents the ratio of actual malware APKs correctly classified to the total number of APKs classified as malware. This metric is important as it predicts the quality of positive predictions made by the model.

$$precision = \frac{TP}{TP + FP}$$

Recall presents the ratio of malware APKs that are correctly classified to the total number of malware APKs that are classified correctly classified as malware or incorrectly as benign. This metric is crucial to understand how accurately our model can recognize relevant data.

$$recall = \frac{TP}{TP + FN}$$

F1-score represents the harmonic mean of precision and recall. This metric helps the researcher to keep a good balance between precision and recall for evaluating the models trained using an imbalanced dataset.

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

From Table 4.1, it is clear that the performance of the XGBoost and Random Forest classifiers was similar and performed better compared to other machine learning classifiers. To understand the performance clearly, the AUC-ROC curve for Decision Tree, XGBoost, and Random Forest are plotted in Fig 4.1. ROC (Receiver Operating Characteristic) curve plots the True Positive Rate against the False Positive Rate, whereas AUC represents the area under this curve. A high AUC score represents that the model accurately distinguishes between malware classes and benign.

The dataset consisted of 525 APK samples of which adware comprised 95 samples, ransomware 93 samples, smsware 108 samples, scareware 111 samples, and benign 118 samples. The class imbalance in the dataset could affect the performance of the model as the class that consists of a large number of samples could decrease the classification of classes that had fewer samples. Thus, it is important to handle the class imbalance problems in the dataset.

In this paper, three approaches to handling class imbalance Synthetic Minority Oversampling Technique (SMOTE), Adaptive Synthetic sampling (ADASYN), and balanced cost were experimented with and analyzed. SMOTE and ADASYN generate new instances for the minority classes to overcome the imbalance problem. The balanced cost approach provides balanced weights to each class so that the imbalance problem is reduced to a fair extent. These three approaches were studied for the top two highest-performing machine learning classifiers, XGBoost, and Random Forest. Table 4.2 represents the evaluation metrics for XGBoost and Random Forest using SMOTE, ADASYN, and Balanced Cost approaches.

From Table 4.2, it could be noticed that the balanced cost approach did not help out XGBoost as the cost applied to the entire tree in one shot which vanishes the cost of classes that had fewer samples [25]. SMOTE approach is sequential in manner and improves the accuracy based on the performance of the previous tree generated had performed better in XGBoost [26]. ADASYN had given average performance on both Random Forest and XGBoost. The AUC-ROC curve for XGBoost using SMOTE and Balanced Cost is presented in Fig 4.3.

Table 4.1 Evaluation Metrics for Machine Learning Classifiers

Classifier	APK Category	Precision	Recall	F1 - Score	Accuracy
Naïve Bayes	adware	0.29	0.47	0.36	0.56
	ransomware	0.88	0.39	0.54	
	smsware	0.48	0.96	0.64	
	scareware	0.79	0.39	0.52	
	benign	0.92	0.57	0.71	
K- Nearest Neighbor	adware	0.33	0.47	0.39	0.65
	ransomware	0.65	0.72	0.68	
	smsware	0.70	0.83	0.76	
	scareware	0.94	0.54	0.68	
	benign	0.67	0.67	0.67	
Support Vector Machine	adware	53	60	56	0.69
	ransomware	67	89	76	
	smsware	84	91	87	
	scareware	71	61	65	
	benign	60	43	50	
Logistic Regression	adware	0.67	0.53	0.59	0.70
	ransomware	0.67	0.78	0.72	
	smsware	0.69	0.96	0.80	
	scareware	0.77	0.61	0.68	
	benign	0.72	0.62	0.67	
Decision Tree	adware	0.62	0.67	0.65	0.78
	ransomware	0.93	0.72	0.81	
	smsware	0.85	0.96	0.90	
	scareware	0.71	0.71	0.71	
	benign	0.81	0.81	0.81	
XGBoost	adware	0.67	0.80	0.73	0.88
	ransomware	1.00	0.89	0.94	
	smsware	0.88	1.00	0.94	
	scareware	0.88	0.75	0.81	
	benign	0.95	0.95	0.95	
Random Forest	adware	0.69	0.73	0.71	0.88
	ransomware	0.89	0.89	0.89	
	smsware	0.88	0.96	0.92	
	scareware	0.92	0.79	0.85	
	benign	0.91	0.95	0.93	

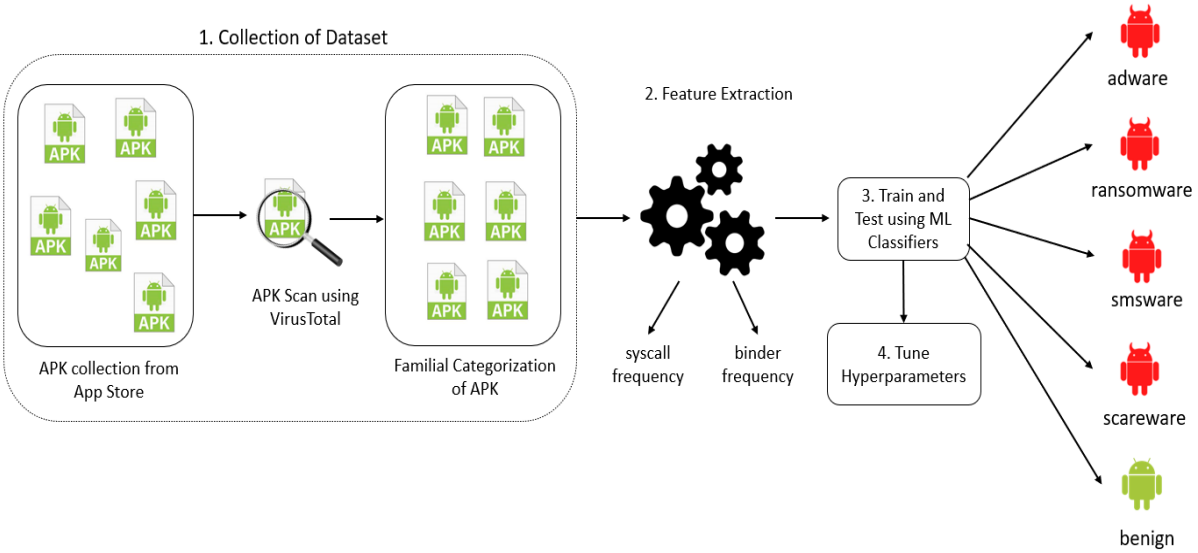

Fig 3.1 Proposed Research Methodology

Table 4.2 Evaluation Metrics for Random Forest and XGBoost using SMOTE, ADASYN, and Balanced Cost

Class Imbalance Approach	Classifier	APK Category	Precision	Recall	F1-Score	Accuracy
Synthetic Minority Oversampling Technique (SMOTE)	Random Forest	adware	0.73	0.73	0.73	0.85
		ransomware	0.81	0.94	0.87	
		smsware	0.88	0.96	0.92	
		scareware	0.88	0.75	0.81	
		benign	0.90	0.86	0.88	
	XGBoost	adware	0.96	0.93	0.95	0.91
		ransomware	0.92	0.92	0.92	
		smsware	0.87	0.93	0.90	
		scareware	0.90	0.86	0.88	
		benign	0.88	0.88	0.88	
Adaptive Synthetic sampling (ADASYN)	Random Forest	adware	0.82	0.93	0.87	0.83
		ransomware	0.76	0.76	0.76	
		smsware	0.80	1.00	0.89	
		scareware	0.87	0.65	0.74	
		benign	0.90	0.78	0.84	
	XGBoost	adware	0.60	0.80	0.69	0.82
		ransomware	0.73	0.89	0.80	
		smsware	0.85	0.96	0.90	
		scareware	1.00	0.64	0.78	
		benign	0.95	0.86	0.90	

Balanced Cost	Random Forest	adware	0.73	0.73	0.73	0.89
		ransomware	0.89	0.94	0.92	
		smsware	0.88	0.96	0.92	
		scareware	0.96	0.82	0.88	
		benign	0.91	0.95	0.93	
	XGBoost	adware	0.77	0.69	0.73	0.81
		ransomware	1.00	0.82	0.90	
		smsware	0.80	1.00	0.89	
		scareware	0.64	0.70	0.67	
		benign	0.91	0.91	0.91	

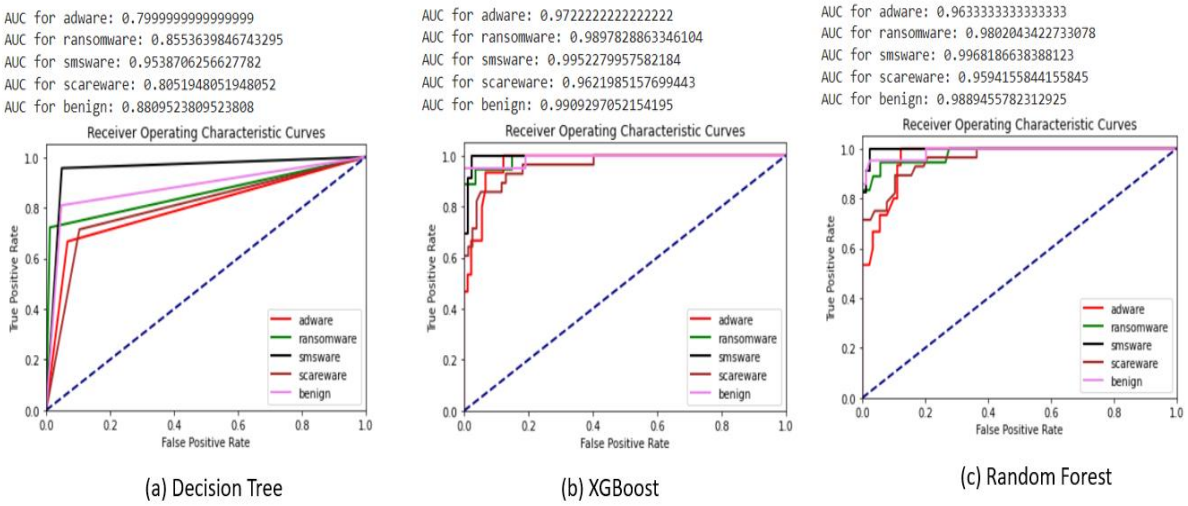

Fig 4.1 AUC-ROC Curve for Decision Tree, XGBoost, and Random Forest

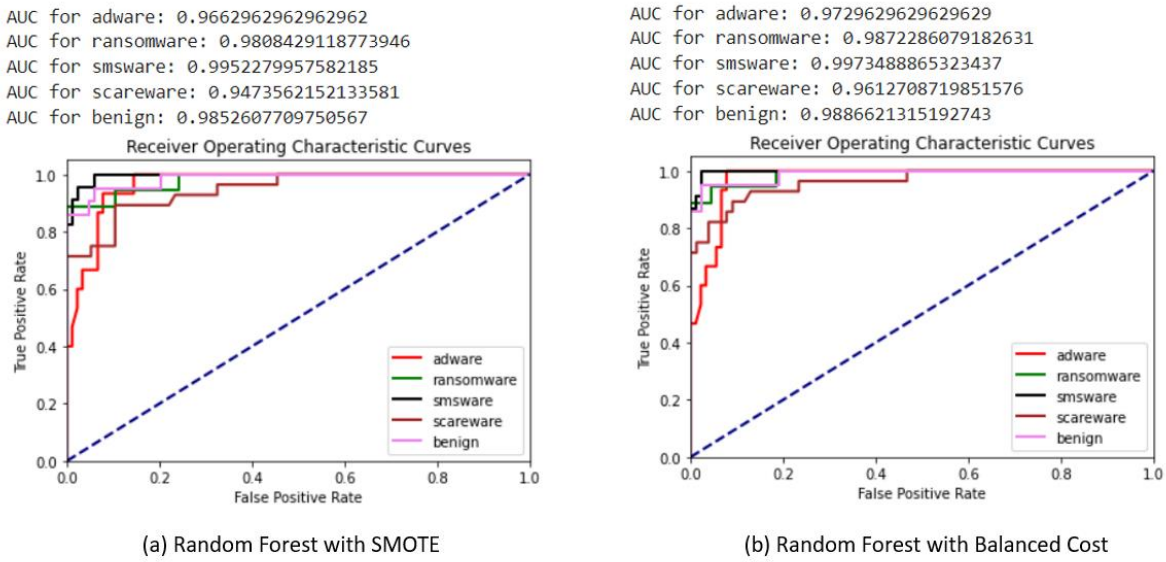

Fig 4.2 AUC-ROC Curve for Random Forest with SMOTE and Balanced Cost Approach

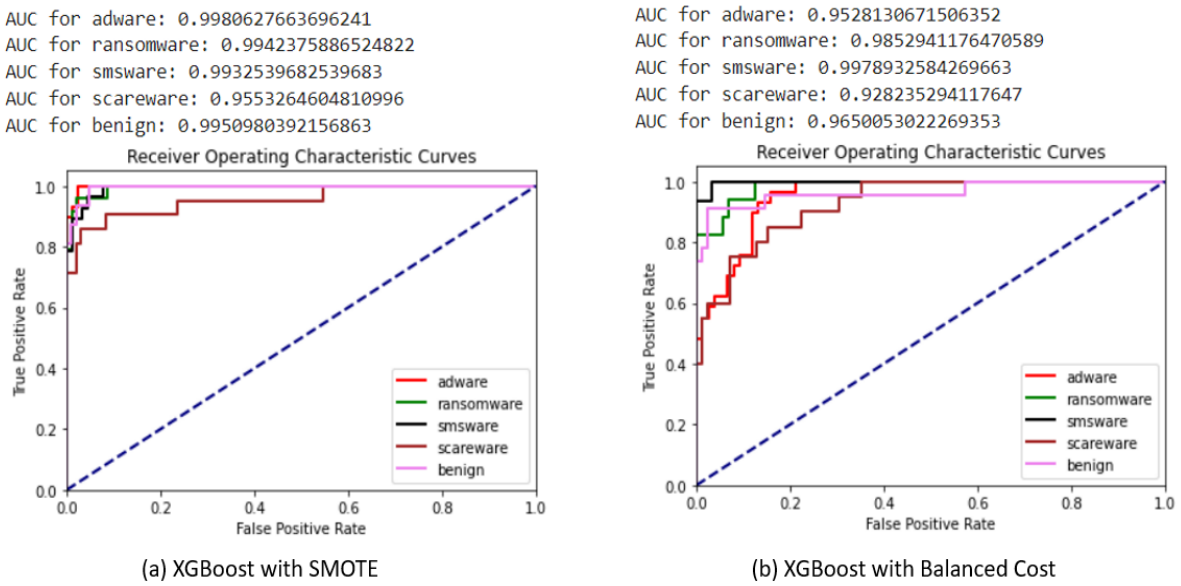

Fig 4.3 AUC-ROC Curve for XGBoost with SMOTE and Balanced Cost Approach

5. Conclusion

We have observed that XGBoost with Synthetic Minority Oversampling Technique (SMOTE) obtained the highest accuracy of 91% in comparison with other machine learning classifiers. It was also noted that the class imbalance affects the accuracy of the machine learning classifiers. Thus, the imbalance problem had to be handled to provide better detection and classification. We also achieved the detection of five families of mobile applications such as adware, ransomware, scareware, smsware, and benign which are the most hazardous malware category in mobile application security. Dynamic features such as syscalls and binders provided resiliency to the security solution developed in this research as these features had to be included by the malware authors in any malware applications to cause havoc in the mobile devices. The dataset collected for this research was recent and thus captured the recentness of the malware behavior.

In the future, we would extend the research by collecting a large set of dynamic features from an Android application and covering various other malware families.

References

[1] "Statcounter - Mobile Operating System Market Share Worldwide." https://gs.statcounter.com/os-market-share/mobile/worldwide

[2] "Statista - Number of smartphone users." https://www.statista.com/statistics/467163/forecast-of-smartphone-users-in-india/

[3] V. Sihag, M. Vardhan, and P. Singh, "A survey of android application and malware hardening," *Comput. Sci. Rev.*, vol. 39, p. 100365, Feb. 2021, doi: 10.1016/j.cosrev.2021.100365.

[4] "AVTEST." https://www.av-test.org/en/statistics/malware/

[5] "SecureList by Kaspersky." https://securelist.com/it-threat-evolution-in-q2-2022-mobile-statistics/107123/

[6] P. E. Black, M. Kass, and E. Fong, "Proceedings of workshop on software security assurance tools, techniques, and metrics," National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 500-265, 2006. doi: 10.6028/NIST.SP.500-265.

[7] S. Hutchinson, B. Zhou, and U. Karabiyik, "Are We Really Protected? An Investigation into the Play Protect Service," in *2019 IEEE International Conference on Big Data (Big Data)*, Los Angeles, CA, USA, Dec. 2019, pp. 4997–5004. doi: 10.1109/BigData47090.2019.9006100.

[8] A. Kiranmayi, N. Padmaja, and P. Student, "Dissecting the Bouncer and Malware Detection in Google Play," vol. 5, no. 8, 2018.

[9] D. Maier, T. Muller, and M. Protsenko, "Divide-and-Conquer: Why Android Malware Cannot Be Stopped," in *2014 Ninth International Conference on Availability, Reliability and Security*, Fribourg, Switzerland, Sep. 2014, pp. 30–39. doi: 10.1109/ARES.2014.12.

[10] L. Wen and H. Yu, "An Android malware detection system based on machine learning," presented at the GREEN ENERGY AND SUSTAINABLE DEVELOPMENT I: Proceedings of the International Conference on Green Energy and Sustainable Development (GESD 2017), Chongqing City, China, 2017, p. 020136. doi: 10.1063/1.4992953.

[11] M. Kakavand, M. Dabbagh, and A. Dehghantanha, "Application of Machine Learning Algorithms for Android Malware Detection," in *Proceedings of the 2018 International Conference on Computational Intelligence and Intelligent Systems*, Phuket Thailand, Nov. 2018, pp. 32–36. doi: 10.1145/3293475.3293489.

[12] B. Amos, H. Turner, and J. White, "Applying machine learning classifiers to dynamic Android malware detection at scale," in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Sardinia, Italy, Jul. 2013, pp. 1666–1671. doi: 10.1109/IWCMC.2013.6583806.

[13] J. D. Koli, "RanDroid: Android malware detection using random machine learning classifiers," in *2018 Technologies for Smart-City Energy Security and Power (ICSESP)*, Bhubaneswar, Mar. 2018, pp. 1–6. doi: 10.1109/ICSESP.2018.8376705.

[14] G. D'Angelo, F. Palmieri, A. Robustelli, and A. Castiglione, "Effective classification of android malware families through dynamic features and neural networks," *Connect. Sci.*, vol. 33, no. 3, pp. 786–801, Jul. 2021, doi: 10.1080/09540091.2021.1889977.

[15] A. Ferrante, E. Medvet, F. Mercaldo, J. Milosevic, and C. A. Visaggio, "Spotting the Malicious Moment: Characterizing Malware Behavior Using Dynamic Features," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*, Salzburg, Austria, Aug. 2016, pp. 372–381. doi: 10.1109/ARES.2016.70.

[16] V. M. Afonso, M. F. de Amorim, A. R. A. Grégio, G. B. Junquera, and P. L. de Geus, "Identifying Android malware using dynamically obtained features," *J. Comput.*

Virol. Hacking Tech., vol. 11, no. 1, pp. 9–17, Feb. 2015, doi: 10.1007/s11416-014-0226-7.

[17] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "EMULATOR vs REAL PHONE: Android Malware Detection Using Machine Learning," in *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, Mar. 2017, pp. 65–72. doi: 10.1145/3041008.3041010.

[18] A. Kapratwar, F. Di Troia, and M. Stamp, "Static and Dynamic Analysis of Android Malware:," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, Porto, Portugal, 2017, pp. 653–662. doi: 10.5220/0006256706530662.

[19] "Contagio Mobile." [Online]. Available: http://contagiominidump.blogspot.com/

[20] "Koodous." [Online]. Available: https://koodous.com/

[21] "APKPure." [Online]. Available: https://m.apkpure.com/

[22] "Google Play Store."

[23] S. Zhu *et al.*, "Measuring and Modeling the Label Dynamics of Online Anti-Malware Engines".

[24] "MobSF." [Online]. Available: https://github.com/MobSF/Mobile-Security-Framework-MobSF

[25] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognit.*, vol. 40, no. 12, pp. 3358–3378, Dec. 2007, doi: 10.1016/j.patcog.2007.04.009.

[26] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem," in *Advances in Knowledge Discovery and Data Mining*, vol. 5476, T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 475–482. doi: 10.1007/978-3-642-01307-2_43.