Journal of Computing and Communication Vol.1 , No.1 , PP. 38-47 , 2022

Using Machine Learning to Identify Android Malware Relying on API calling sequences and Permissions

Eslam Amer*[a], Seif ElDein Mohamed[a], Mostafa Ashaf[a], Amr Ehab[a], Omar Shereef[a], Haytham Metwaie[a], Ammar Mohammed[b]

[a]Faculty of Computer Science - Misr International University - Cairo - Egypt
[b]Faculty Of Computer Science - Misr International University, Cairo, Egypt (on Leave) Cairo University

*Corresponding author: Eslam Amer [*eslam.amer@miuegypt.edu.eg*]

ARTICLE DATA

Article history:
Received 28 Dec 2021
Revised 02 Feb 2022
Accepted 08 Feb 2022
Available online

Keywords:
Malware Detection
API call sequence
Permission

ABSTRACT

The revolutionary in cyber attacks, especially in smartphones are rising. The Android operating system is becoming one of the most leading operating systems. Therefore, Android malware is rising in terms of popularity. Malware makers are using novel techniques to develop malicious Android applications, drastically diminishing the capabilities of traditional malware detectors. In consequence, those Anti-malware detectors become unable to detect these unexplained malicious apps. Currently, machine learning techniques are extensively used to discover new unknown Android viruses by analyzing the functionality of static and dynamic app reviews. In this paper, we introduce an Android malware detection technique based on API and permissions. Our purpose is to evaluate and examine the incorporation of machine learning classifiers with featured Android features such as APIs and permissions. We investigated several classification methods in characterizing Android malware with respect to the used feature. We discovered varied performance when we analyses all Android malware detection classifiers that use machine learning, suggesting that machine learning algorithms are effectively utilized in this area of identifying Android malicious apps.

1. Introduction

Currently, most of the current daily tasks are entwined inexorably with our cell phones. It is shown that since 2012 to 2021, Android kept its leading position as the world's widely used mobile operating system, commanding the mobile OS market with a 71.93% share [1] as shown in Fig. 1.

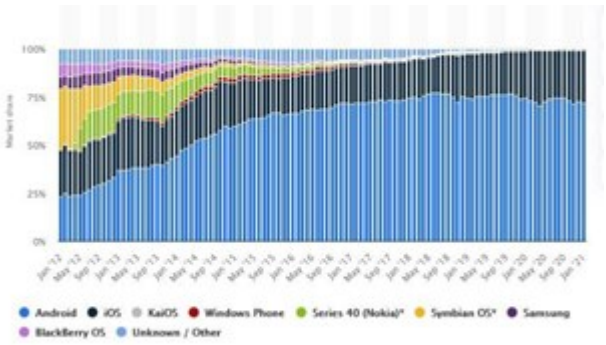

Figure 1: Widespread mobile operating system during 2012-2021

Moreover, along with the continual development of Android OS, cybercriminals are getting more skilled and inventive. Throughout their skillful techniques, new and complex varieties of malware are appearing which cause malware detection to be increasingly challenging. Accordingly, malware analysis, which entails determining the origins, functionality, and potential consequences of every malware sample, is critical in today's world of cybersecurity [2].Nowadays, Android apps have gained

more popularity than other applications. There are over one million Android apps of different domains, like WeChat, Tik-Tok, and mobile banking, which are usually used on a regular basis [3, 4].

Those applications are continue to play an increasingly prominent role in Android platforms like as Google Play. However, the great majority of these apps have exposed users' sensitive information, such as their location, payment cards information, and contact information.

Almost all programmes have access to the users' private information, despite the fact that this allows for more tailored administrations. Additionally, it might result in the leakage of sensitive information and financial distress. Moreover, a recent cyber assault, the ransomware "CovidLock," exemplifies that private information. This ransomware version attacks their targets using malicious apps ostensibly containing further information about the condition. CovidLock encrypts data saved on Android smartphones and prevents victims from accessing it once activated. To access the encrypted data, victims must pay a ransom of USD 100 per machine. Additionally, Android malware apps continue to grow at a breakneck pace. This security concern has been gaining traction in both the commercial and academic worlds.

Numerous studies on malware have been undertaken. Dynamic and static analysis are the two basic types of identification procedures available at the moment. Each strategy offers a number of advantages and disadvantages. Techniques of static analysis like, Apriori [5], and DREBIN [6] without running them, analyses programmes. Regardless, the solutions do not provide protection against anti-decompilation and obfuscation. On the other side, dynamic analysis tools, such as VetDroid, [7] continuously monitor programmes for harmful activity. Nonetheless, it is difficult to record every execution activity. Due to the rapid development of malware, adaptive machine learning methods are used to conduct Android harmful detection. As a consequence, gathering machine learning attributes that best depict criminal behavior assists in evaluating malware regions.

2. Malware Taxonomy

Malware operations are heavily reliant on obtaining users' sensitive data via theft, surveillance, and the display of unwanted advertisements. Malware is a subset of malicious software, and it is generally defined as a piece of software that knowingly embraces the core characteristics of malware aggressors and articulates its harmful intent. Fig. 2 illustrates many types of malware depending on their distinct objectives and techniques of penetration.

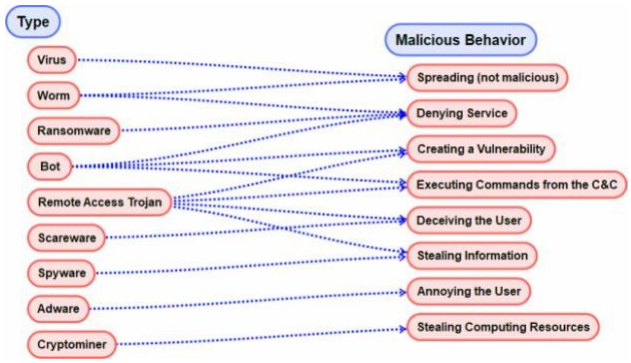

Figure 2: Types of malware

Virus: A kind of computer programs which must be activated by their host's activation to perform its malicious activity.

Worms: Worms are self-replicating harmful programs that may spread autonomously after they have gained access to a system. In contrast to viruses, worms do not need activation by their hosts.

Ransomware: The brand new type of malware attacks. Ransomware encrypts users data and restrict their access to it.

Bots: Numerous devices linked to the Internet, each of which hosts one or more bots. Botnets may be used to launch Distributed Denial-of-Service attacks, steal data, transmit spam, and provide the attacker access to the device and its network.

Trojan Horses: Inspired by the ancient Greek legend of the Trojan Horse deception ultimately led to the fallen of the city of Troy.Trojan Horse is a malicious programms that masquerades as a legitimate type of programs or applications, however, they conceals harmful code.

Scareware: A type of malicious attacks that convinces people to download or purchase hazardous and often meaningless software. Often produced using scareware, pop-up adverts prey on users' paranoia in order to affect their behaviour.

Spyware: The spyware applications is considered as malicious software that sneak into users computers and steals information about their internet activities and their personal data.
Adware: A kind of malware applications that intended to show advertisements on the screen of your computer based on the analysis of users behaviour.
Cryptominer: Which is considered an unlawful mining of cryptocurrencies on users' personal computers.

Recent research [8, 9] studies assert that detection methodologies that employ machine learning or "anomaly detection" have become the leading and most reliable methods in recognizing Android malware. In contrast to static analysis, which implies the manual examination of the AndroidManifest.xml file, dynamic analysis methods require monitoring the interaction behavior of an application with the operating system in real-time. Therefore, malicious or non-malicious applications are executed in a secured, isolated environment. The role of machine learning methods is to extract, find, and collect general behavioral patterns that differentiate maliciously versus non-malicious behaviors. Hence, those machine learning models can identify new malicious attacks.

3. Background

Malware analysis is used by security experts for several applications. The main application of malware analysis is to study and evaluate the situation when malware attacks occur. Moreover, to find out the type and category of malware that is/are involved in the attack. Similarly, a deep grasp of the features and impacts of every malware sample helps them to repel cyber threats more effectively. The majority of malware analysis approaches are classified as static or dynamic [10, 11, 12, 13].

Static malware analysis is a technique for examining malware samples that have not been launched or executed. In contrast, dynamic malware analysis involves studying the malicious code through lunching it in a secured, controlled environment. The malicious samples are executed inside a secure, isolated virtual environment called sandbox, where the malware activities are monitored [12, 13].

Faiz [14], proposed a method for detecting Android malware apps with a hybrid classification approach utilising the K-means clustering technique and a support vector machine (SVM). The researchers analysed two kinds of data. [15] [16].They create two datasets, Data1 and Data2, from the first one. The data set contains 13,176 training applications and 1860 test applications. Data2 is made up of 12,028 training applications and 3008 test apps. The second dataset includes 230 app-pairs that collaborate. They think that cooperative app-pairs will be responsible for a large portion of the harm caused by Android malware. They then use the parameter vector and a simple judgement algorithm to identify application collusion.

Bhat et al [17], The framework uses a naïve Bayes model to identify fraudulent Android apps. They obtained two datasets including 2870 applications: DREBIN [15] and PRAGuard [18] They obtained 1472 harmful apps and 1398 benign applications from 2870 applications. The researchers address the issue by using a malware detection tool known as MaplDroid, which combines static analysis with naive Bayes model analysis. Additionally, they conducted comparison research using alternative categorization approach, such as random forest. MaplDroid was successfully achieved a score of 99.12%.

Jannat et al., [19] developed a system that uses machine learning to analyse and identify malware on Android. The researchers approach the issue in two distinct ways: via dynamic and static analysis. They get the greatest dynamic analysis result by using the Random Forest (RF) technique, which is an enlarged form of the decision tree (DT) algorithm. Dynamic analysis's accuracy ratings exceeded those of static analysis by more than 93%. Additionally, the researchers conducted static and dynamic analyses on a variety of datasets. In their work, they conducted static analysis on the MalGenome dataset. It comprises around 360 harmful programmes grouped by malware family and another dataset from Kaggle including 4000 dangerous applications in JSON format [20]. Furthermore, they employed a MalGenome dataset for dynamic analysis, which comprises 1260 harmful software programmes classified as belonging to 49 unique malware families.

Zhuo Ma [21] suggested a technique for detecting malicious Android apps. Control flow graphs and machine learning methods were employed by the researchers. They create historical datasets and train them using a recurrent neural network approach called Long short-term memory. The researchers disassembled and configured three distinct types of systems: API use datasets, API frequency datasets, and API sequence datasets. They then attain a detection accuracy of 98.98 percent.

Muhammad Murtaz [22] recommended the development of a mechanism for identifying harmful Android applications. They address this problem by using six approaches to the knowledge analysis dataset they obtain: K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Decision Tree (J48), Neural Networks (NN), Naive Bayes (NB), and Random Forest (RF). They used the CICAndMal2017 dataset, which comprises 10854 records (6500 benign and 4354 malicious), and the dataset is divided into four intriguing groups (Adware, Ransomware, Scareware, SMS Malware). They demonstrate in this study that they can identify malware locations in nine different motions, hence increasing the productivity of activity classifiers. Additionally, the model utilises collecting approaches such as stream-based, bundle-based, and time-based highlights to define malware families. The evaluation demonstrates that the suggested feature set has over 94 components critical for real-world malware recognition algorithms.

Utama et al. [23], suggested a method for analysing and determining the danger level of Android applications. The researchers employed the Naive Bayes (NB) method to determine whether or not the android application was malicious. They utilised a dataset with 188,389 records. They studied the permissions and vulnerabilities discovered in this investigation. Additionally, they can determine if an android application is safe or not using this approach. Finally, they determine that this study is 97.2 percent accurate.

Yan et al. [24], investigated a system that identifies malicious Android applications with machine learning. The researchers acquired two datasets: one on the malgenome and another on viral sharing. Additionally, they downloaded and evaluated over 1,000 applications from Google Play. They overcome this limitation by collecting runtime logs from each application and use the data retrieved from the logs. The researchers achieved a result that had a false-positive rate less than 8% and a real positive rate greater than 90%.

Rashidi et al. [25] suggested a method for identifying fraudulent Android applications that uses support vector machine (SVM) and active learning. The authors used their bespoke instrumentation system, DroidCat, to record a log of apps. They utilised scanning and parsing algorithms after collecting logs. They made use of a dataset dubbed the Drebin Project. The dataset contains over 5000 apps from 179 malware families; they chose 500 harmful and 500 benign applications for training with the RBF (Radial basis function) Kernel and 200 malicious and 200 benign applications for testing using the researchers' model.

The running time for each application was estimated to be between 2 and 5 minutes, which equated to 79 hours for all apps. Their exploratory results demonstrate that their suggested approach achieves acceptable true-positive and false-positive rates and automatically modifies the identification model to accommodate new virus types. Additionally, they applied the Quantitative Query Strategy (QQS), which has a significant effect on the model's performance and accuracy.

Yuxia et al. [26], on the basis of excessive learning, they presented a solution for identifying malware in Android apps. The researchers gathered data from Tencent's YingYongBao online shop. They tested them using 524 benign and 525 malicious programmes. To differentiate fraudulent Android applications, they got the application's permissions and API requests through excessive learning models. Their findings demonstrated that they outperform existing human intrusion , effectiveness in detection efficiency.

Lageman et al. [27] used the controlled flaw in a device which is capable of identifying malicious Android applications based on its runtime behaviour. In their work, they created runtime datasets by analysing logs and trace output. They obtained the data from the Android Malware Genome project at North Carolina State University. They used both the Random Forest (RF) and support vector machine (SVM) classifiers to test the dataset. Their findings showed that the RF classifier outperformed the SVM with 90% true-positive rate and a false-positive rate of almost 6%.

4. The Proposed Approach

Our proposed system, as illustrated in Fig.3 contained two main phases namely, preprocessing, and processing. We viewed the malware detection challenge as a type of a classification problem using machine learning. Our input consists of 3800 unique Android applications that were gathered from the Malgenome data set. The Malgenome dataset contains permissions and API calls for benign and malicious apps, likewise the Maldroid dataset, which contains innocuous apps, adware, banking malware, and mobile riskware.

Pre-processing entails collecting features from our apps by acquiring their most frequently used API calls and permissions in order to lower the dimensionality of the raw data for our model, using less compute power, and preparing the appropriate features for training and testing. If required, the data set will be normalised, standardised, and anomaly detected.

We split our data set into three subgroups throughout the processing stage: training, validation, and testing. We will use the training set to train and test multiple models; our malware detection model will be trained and evaluated on the data set using a variety of methods, including K-nearest neighbour, Naive Bayes, Support vector machine, and Decision tree. We will utilize dynamic analysis to monitor system API calls and permissions throughout an application's execution and will report the findings to a log file.

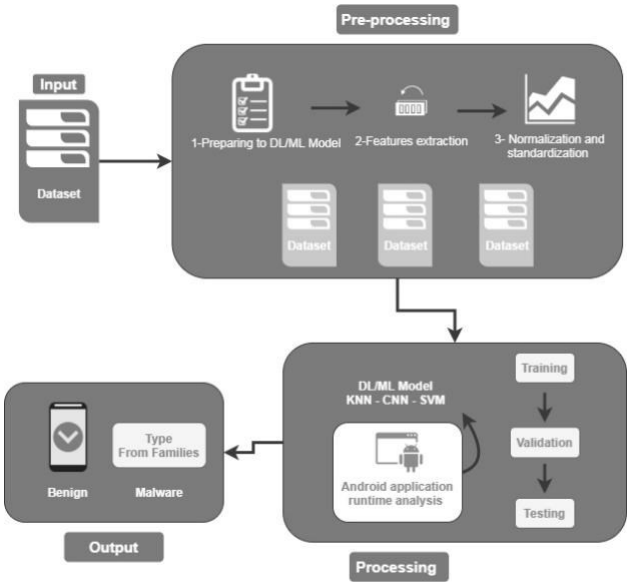

Figure 3: The Proposed Malware Analysis Framework

After scanning all programmes, we will identify each chosen application as benign or as belonging to one of nine malware families (viruses, worms, ransomware, spyware, etc.) based on their permissions and API calls, as well as detecting its dynamic behaviour.

4.1. Dataset

The data sets upon which our model is built are referred to as Malgenome data set [28] and Maldroid data set [29]. The Malgenome data collection contains 1260 malware programmes and 2539 benign applications; it also contains 3800 applications with a set of permissions and API calls for each application, as well as a list of the most frequently occurring permissions and API calls for malware and benign applications. The Maldroid data set comprises 11599 programmes categorised as benign, adware, banking malware, and mobile riskware.

4.2. Evaluation Metrics

We used several standard evaluation metrics to evaluate the performance of our model. The most standard metrics include, accuracy (ACC), precision (PREC), sensitivity (recall) (REC), specificity, and f-score (F1). They are calculated as follows:

$$Acc = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

$$prec = \frac{TP}{TP+FP} \quad (2)$$

$$Rec = \frac{TP}{TP+FN} \quad (3)$$

$$prec = 2 * \frac{Prec*Rec}{PrecRec} \quad (4)$$

Where TP, TF, FP, and FN indicate the true positive, true negative, false positive, and false negative, respectively.

5. Results and Discussion

The Malgenome and Maldroid data sets were used in our experiments, and the results were promising. Using the Python programming language, we created four different machine learning models to compare and contrast. Among the classifiers we used were the KNN, the NB, the SVM with linear kernel, and the Decision Tree Classifier. The naive Bayes model was rejected from the Maldroid data set because of its very low accuracy of 51 percent. Because they have a greater degree of accuracy and precision, we employ them, notably the linear kernel with SVM algorithm, which has an accuracy of 0.86 percent on the Maldroid data set and a precision of 99.9 percent on the Malgenom data set, among other advantages.

Table 1
Accuracy results over Malgenom API calls Dataset

Classifier	Accuracy	Precision		Recall		f1-score	
		Malware	Benign	Malware	Benign	Malware	Benign
KNN	0.973	0.981	1.000	1.000	1.000	1.000	1.000
Naive Bayes	1.000	1.000	1.000	1.000	1.000	1.000	1.000
SVM	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Decision Tree	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table 2
Acuracy results over Malgenom Permissions Dataset

Classifier	Accuracy	Precision		Recall		f1-score	
		Malware	Benign	Malware	Benign	Malware	Benign
KNN	0.993	0.982	1.000	1.000	1.000	1.000	1.000
Naive Bayes	0.974	0.992	1.000	1.000	1.000	1.000	1.000
SVM	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Decision Tree	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table 3
Maldroid Dataset Accuracy Results using API calls

Classifier	Accuracy	Precision					Recall					f1-score				
		Ben	Adware	Bank mal	SMS mal	Mobrisk	Ben	Adware	Bank mal	SMS mal	Mobrisk	Ben	Adware	Bank mal	SMS mal	Mobrisk
KNN	0.852	0.741	0.712	0.851	0.981	0.832	0.752	0.692	0.882	0.932	0.891	0.753	0.701	0.872	0.951	0.862
SVM	0.861	0.982	1.000	0.622	0.961	0.771	0.981	0.991	0.912	0.742	0.961	0.981	1.000	0.742	0.842	0.853
Decision Tree	0.881	1.000	0.000	1.000	1.000	1.000	1.000	1.000	0.622	1.000	1.000	1.000	0.000	0.762	1.000	1.000

Initially, Malgenom was used for training, testing, and validation of API requests and application permissions. The initial training set is given to the models to update their parameters, and then 20% of the remaining model is tested to determine whether an app is benign or malicious.

Secondly, Maldroid uses training, testing, and validation to categorise benign apps, adware, banking malware, and mobile risk ware applications. The pre-trained model is then evaluated against the testing set to ensure correctness. We achieved a high level of accuracy with the Malgenom data set and a moderate level of accuracy with Maldroid

since we removed the naïve Bayes method. The tables 1, 2, and 3 describe the accuracy of the four classifiers on both data-sets.

Figures 4, 5, 6, 7, illustrated the most frequently used permissions and API calls along with their occurrence frequency in Malgenome data set. The malicious permissions and APIs are plotted in red while the blue plotting refers to the benign ones.

Although our proposed model returned promising results, we are completely convinced that our approach requires a continual update in terms of training. The reason is that the new malware variants or adversarial attacks can identify the training patterns, and therefore they can skip or fake the detection model. Accordingly, we shouldn't be faked by our accuracy in the long term because the rapid malware development also comes up with new tricks and patterns that may be new to our model. Therefore, our proposed model should be continually enriched with new samples that enable it to be up-to-date.

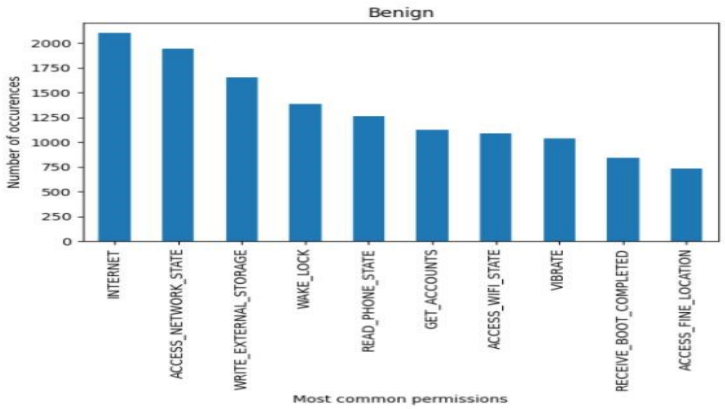

Figure 4: Common Benign Permissions

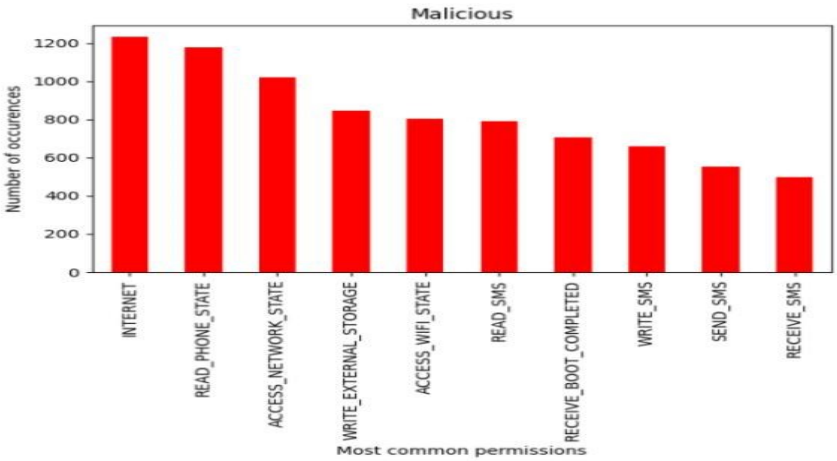

Figure 5: Common Malicious Permissions

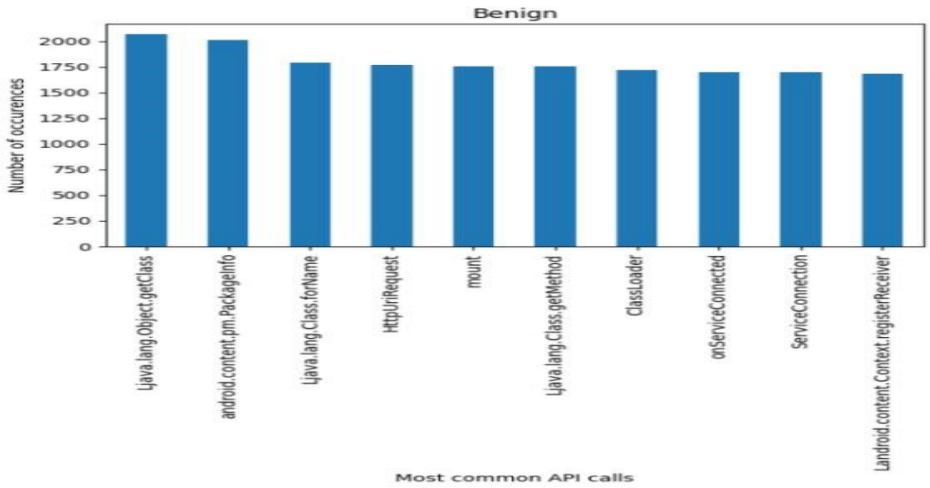

Figure 6: Common Benign API calls

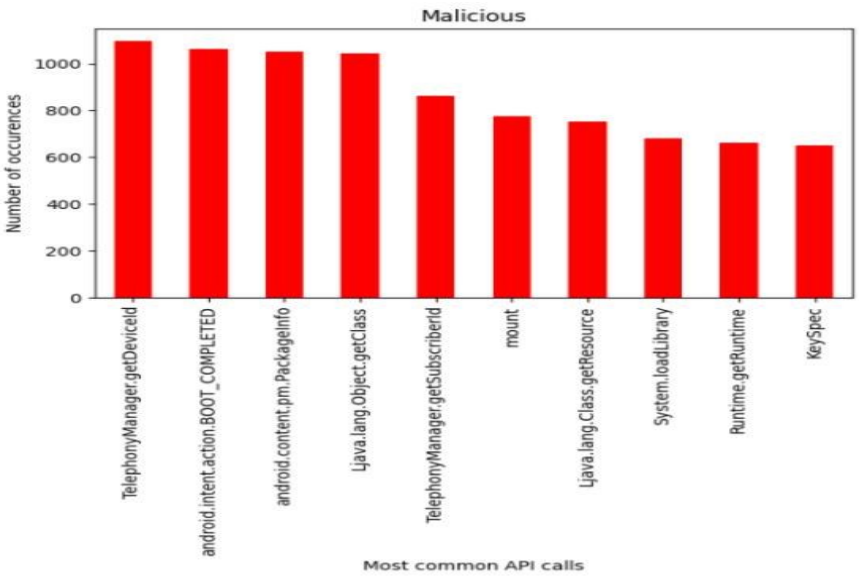

Figure 7: Common Malicious API calls

6. CONCLUSIONS AND FUTURE WORK

The instantaneous increase in the number of dangerous programs nowadays necessitates dependable malware detection technologies to keep up. Our research discovered particular characteristics of our suggested model that were used to estimate detection efficiency. Our experiments used two separate datasets for our experiments, which were

successful. According to our findings, the suggested technique achieved a high level of accuracy with the Malgenome data set, with an average of 99 percent, and a lesser level of accuracy with the Maldroid data set, with an average of 86 percent, according to our findings. Our choice algorithms were k-nearest neighbor, support vector machine, naive Bayes, and decision tree, all of which were chosen for their high accuracy and fast processing times. We intended to do a more in-depth investigation of feature selection in future work with a larger data set. Based on their dynamic behavior and API calls and permissions, this paper provides a technique for spotting rogue Android applications in the wild. Developing a mobile application that will operate in concert with our machine learning model to scan all apps for malware and benign code is planned for the near future.

References

[1] statista. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/, 2021.

[2] Eslam Amer. Permission-based approach for android malware analysis through ensemble-based voting model. In 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), pages 135–139. IEEE, 2021.

[3] Mobile App Statistics To Know In 2020. https://mindsea.com/app-stats/, 2020.

[4] Seif ElDein Mohamed, Mostafa Ashaf, Amr Ehab, Omar Shereef, Haytham Metwaie, and Eslam Amer. Detecting malicious android applications based on api calls and permissions using machine learning algorithms. In 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), pages 1–6. IEEE, 2021.

[5] Oscar Somarriba, Urko Zurutuza, Roberto Uribeetxeberria, Laurent Delosières, and Simin Nadjm-Tehrani. Detection and visualization of android malware behavior. Journal of Electrical and Computer Engineering, 2016, 2016.

[6] Xiang Li, Jianyi Liu, Yanyu Huo, Ru Zhang, and Yuangang Yao. An android malware detection method based on androidmanifest file. In 2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS), pages 239–243. IEEE, 2016.

[7] Yuan Zhang, Min Yang, Bingquan Xu, Zhemin Yang, Guofei Gu, Peng Ning, X Sean Wang, and Binyu Zang. Vetting undesirable behaviors in android apps with permission use analysis. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pages 611–622, 2013.

[8] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. Yes, machine learning can be more secure! a case study on android malware detection. IEEE Transactions on Dependable and Secure Computing, 16(4):711–724, 2017.

[9] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni. Survey of machine learning techniques for malware analysis. Computers & Security, 81:123–147, 2019.

[10] Ivan Zelinka and Eslam Amer. An ensemble-based malware detection model using minimum feature set. In MENDEL, volume 25, pages 1–10, 2019.

[11] Eslam Amer, Shaker El-Sappagh, and Jong Wan Hu. Contextual identification of windows malware through semantic interpretation of api call sequence. Applied Sciences, 10(21):7673, 2020.

[12] Eslam Amer and Ivan Zelinka. A dynamic windows malware detection and prediction method based on contextual understanding of api call sequence. Computers & Security, 92:101760, 2020.

[13] Eslam Amer, Ivan Zelinka, and Shaker El-Sappagh. A multi-perspective malware detection approach through behavioral fusion of api call sequence. Computers & Security, 110:102449, 2021.

[14] Md Faiz Iqbal Faiz and Md Anwar Hussain. Hybrid classification model to detect android application-collusion. In 2020 43rd International Conference on Telecommunications and Signal Processing (TSP), pages 492–495. IEEE, 2020.

[15] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In Ndss, volume 14, pages 23–26, 2014.

[16] Static Analysis of Android Malware. https://www.kaggle.com/goorax/static-analysis-of-android-malware-of-2017.

[17] Parnika Bhat, Kamlesh Dutta, and Sukhbir Singh. Mapldroid: Malicious android application detection based on naive bayes using multiple. In 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT), pages 49–54. IEEE, 2019.

[18] Android PRAGuard Dataset. http://pralab.diee.unica.it/en/androidpraguarddataset, 2018.

[19] Umme Sumaya Jannat, Syed Md Hasnayeen, Mirza Kamrul Bashar Shuhan, and Md Sadek Ferdous. Analysis and detection of malware in android applications using machine learning. In 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), pages 1–7. IEEE, 2019.

[20] Android Malware Genome Project. http://www.malgenomeproject.org/.

[21] Zhuo Ma, Haoran Ge, Yang Liu, Meng Zhao, and Jianfeng Ma. A combination method for android malware detection based on control flow graphs and machine learning algorithms. IEEE access, 7:21235–21245, 2019.

[22] Muhammad Murtaz, Hassan Azwar, Syed Baqir Ali, and Saad Rehman. A framework for android malware detection and classification. In 2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS), pages 1–5. IEEE, 2018.

[23] Ridho Alif Utama, Parman Sukarno, and Erwid Musthofa Jadied. Analysis and classification of danger level in android applications using naive bayes algorithm. In 2018 6th International Conference on Information and Communication Technology (ICoICT), pages 281–285. IEEE, 2018.

[24] Hongbing Yan, Yan Xiong, Wenchao Huang, Jianmeng Huang, and Zhaoyi Meng. Automatically detecting malicious sensitive data usage in android applications. In 2018 4th International Conference on Big Data Computing and Communications (BIGCOM), pages 102–107. IEEE, 2018.

[25] Bahman Rashidi, Carol Fung, and Elisa Bertino. Android malicious application detection using support vector machine and active learning. In 2017 13th International Conference on Network and Service Management (CNSM), pages 1–9. IEEE, 2017.

[26] Yuxia Sun, Yunlong Xie, Zhi Qiu, Yuchang Pan, Jian Weng, and Song Guo. Detecting android malware based on extreme learning machine. In 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), pages 47–53. IEEE, 2017.

[27] Nathaniel Lageman, Mark Lindsey, and William Glodek. Detecting malicious android applications from runtime behavior. In MILCOM 2015-2015 IEEE Military Communications Conference, pages 324–329. IEEE, 2015.

[28] Xuxian Jiang Yajin Zhou. http://www.malgenomeproject.org/, 2015.

[29] CICMalDroid 2020. https://www.unb.ca/cic/datasets/maldroid-2020.html/, 2020.