2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC) | 978-1-6654-1243-8/20/$31.00 ©2021 IEEE | DOI: 10.1109/MIUCC52538.2021.9447594

Detecting Malicious Android Applications Based On API calls and Permissions Using Machine learning Algorithms

Seif ElDein Mohamed, Mostafa Ashaf, Amr Ehab, Omar Shereef, Haytham Metwaie, Eslam Amer
Misr International University
Faculty of Computer Science
Cairo, Egypt
{*seifeldein1702622, mostafa1710792, amr1708665, omar1704459, haytham.metwaie, eslam.amer*}*@miuegypt.edu.eg*

***Abstract*—Android malware is growing, and the Android operating system is becoming more mainstream. Malware developers are using new strategies to build harmful Android apps, significantly weakening the capability of conventional malware detectors, which are unable to identify these mysterious malicious applications. Machine learning methods can be used to identify unknown Android malware using the functionality gleaned from static and dynamic reviews of Android apps. This article aims to compare and analyze different Android malware detection systems based on detection techniques, analysis processes, and extracted features. We learned scientific investigations in all Android malware detection approaches that use machine learning, demonstrating that machine learning algorithms are often used in this area to identify Malicious programs in the wild.**

I. Introduction

Our everyday lives are inextricably linked to our smartphones. And the consumer share of mobile platforms worldwide from 2012 to 2021, Android retained its leadership status as the world's largest mobile operating system in January 2021, controlling the mobile OS market with a 71.93 % share [1] as shown in fig. 1

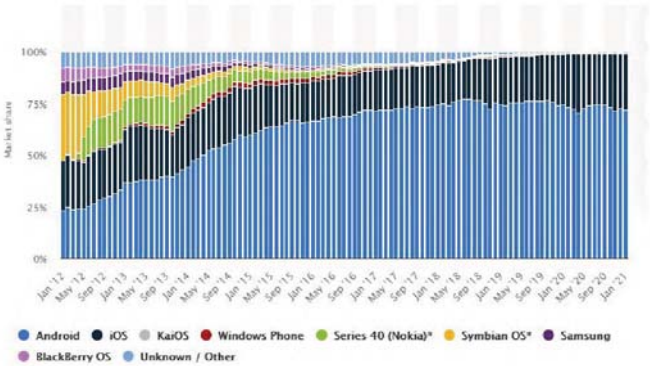

Fig. 1: Most popular mobile operating system during 2012-2021

Cybercriminals are becoming more sophisticated and imaginative, new and advanced types of malware are emerging, and malware detection is becoming increasingly difficult. Malware analysis, which includes analysing the sources, functionalities, and possible effect of any malware sample, is crucial in today's world of cybersecurity.

More than one million Android applications such as WeChat, Tik-Tok, and mobile banking applications are commonly used daily [2], which keep on playing a progressively significant role found in Android markets like Google Play. The vast majority of these applications have breached the users' private data, for example, their location, credit cards, and contacts data.

Practically all applications can access the users' private information, even though this gives users better-customized administrations. It might likewise result in data spillage of private information and financial misfortune. And a late malware attack, the ransomware "CovidLock" is an exemplification. This variant of ransomware infects targets through malicious programs that purport to include additional knowledge about the disease. The issue is that once enabled, CovidLock encrypts data stored on Android devices and prevents victims from accessing it. Victims have to pay a ransom of USD 100 per computer to unlock the decrypted data. Moreover, Android malware applications continue rising perpetually. This security issue has been broadly expanding in the business and academic fields.

A huge group of researches against malware has been conducted. As of now, the two primary kinds of identification techniques are dynamic and static analyses. Each approach has benefits and deficiencies. Static analysis techniques, for example, PApriori [3], and DREBIN [4] examine applications without executing them. Be that as it may, the strategies cannot protect against anti-decompiling and obfuscation. On the other hand, dynamic analysis techniques, for example, VetDroid, [5] run the applications in real-time to detect malicious behavior. Yet, it is hard to capture all the execution behavior. With malware being quickly developing, adaptive machine learning techniques are utilized to perform Android malicious detection. As a result, collecting features that best reflect malicious activity as machine learning features aids in analyzing malware areas.

978-1-6654-1243-8/21/$31.00 ©2021 IEEE

II. Malware Taxonomy

Malware operations are extremely referenced to getting to users' private data by stealing, spying, and showing regrettable ads. Malware is included within malicious software, and it is frequently indicated as a software program that consciously owns the deep attributes of malware aggressors and describes its malicious aim. Various kinds of malware are depicted in Fig. 2 based on their different purposes and infiltration methods.

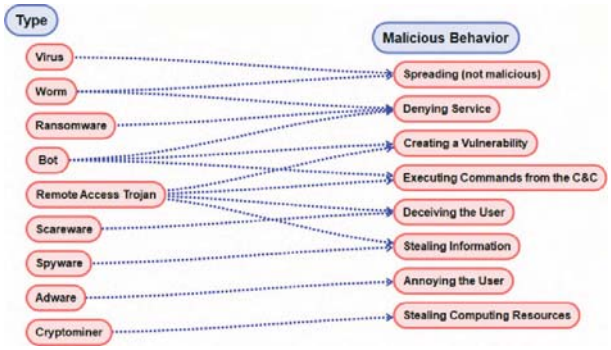

Fig. 2: Types of malware

- **Virus:** Viruses, unlike worms, require an infected working operating device or software to function.
- **Worms:** Worms are spread via software vulnerabilities or phishing attacks.
- **Ransomware:** Ransomware prevents or limits access to your own data.
- **Bots:** A bot is a device which has been compromised to make the machine part of a network controllable to a hacker
- **Trojan Horses:** a Trojan Horse is a malicious application that seems to be a regular piece of software but hides something malicious.
- **Scareware:** Scareware is a malware technique that dupes users into downloading or purchasing malicious and often pointless applications. Sometimes created using scareware, pop-up ads take advantage of users' paranoia to influence their behavior.
- **Spyware:** Spyware is unwanted software that infiltrates your computing device, stealing your internet usage data and sensitive information.
- **Adware:** Adware is a malicious malware that is programmed to display ads on your computer screen.
- **cryptominer:** Cryptojacking is the unauthorized use of someone else's computer to mine cryptocurrency.

III. Related Work

Security professionals use malware analysis for a variety of purposes. They may use it to determine the degree of infection after a malware strike or to classify the type of malware involved. Similarly, a thorough understanding of any malware sample's functionalities and effects enables them to better combat cyber attacks. Malware analysis techniques are mainly categorized into either static or dynamic [6]–[8].

Static malware analysis is the method of analyzing a malware sample without running or executing it. On the other hand, dynamic malware analysis includes analyzing the code as it is running in a controlled environment. The malware is run in a locked, isolated virtual environment, and its behavior is observed [8].

Faiz [9], introduced a system that detects android malware applications by hybrid classification with K-means clustering algorithm and support vector machine (SVM). The researchers used two datasets [10] [11]. From the first dataset, they generate two datasets Data1 and Data2. Data1 consists of 13,176 applications to train the model and a test set of 1860 applications. Data2 consists of 12,028 applications to train the model and a test set of 3008 applications. The second dataset consists of 230 colluding app-pairs. They believe the cooperating app-pairs will carry out many of the dangers carried out by Android malware. They then detect application collusion using the parameter vector and a basic judgment algorithm.

Bhat et al [12], framework detects malicious android applications based on naive Bayes model. They got two datasets DREBIN [10] and PRAGuard [13] datasets that contain 2870 applications. From 2870 applications, they got 1472 malicious applications and 1398 benign applications. The researchers solve the problem using a malware detection technique called MaplDroid uses a static analysis approach with naive Bayes model analysis. Also, they applied another classification technique, such as random forest, to bring out comparative analysis. MaplDroid achieved a score of 99.12%.

Jannat et al., [14] exploited a system that does analysis and detection of malware in android using machine learning. The researchers solve the problem in two ways by dynamic analysis and static analysis. They get the best result in the dynamic analysis using the Random Forest (RF) algorithm, which is an expanded variant of the decision tree (DT) algorithm. The accuracy scores of dynamic analysis surpassed those of static analysis by more than 93%. Furthermore, the researchers used various datasets for static and dynamic studies.

They used the MalGenome dataset for static analysis. It consists of around 360 applications assembled by their malware families and another dataset from Kaggle that contains 4000 malicious applications in JSON format. And in dynamic analysis, they used a MalGenome [15] dataset that contains 1260 malicious software applications listed as relating to 49 distinct malware families.

Zhuo Ma [16], proposed a system that makes a detection for malware android applications. The researchers used control flow graphs and machine learning algorithms. They build chronological datasets and train them by making a Long short-term memory algorithm which is a recurrent neural network. The researchers de-compile and set up 3 kinds of systems: API usage datasets, API frequency datasets, and API sequence datasets. Then, they achieve 98.98% detection precision.

Muhammad Murtaz [17] proposed a framework that scope to detect android malware applications. They solve the problem by using 6 algorithms K-Nearest Neighbor (KNN), Sup-

port Vector Machine (SVM), Decision Tree (J48), Neural Networks (NN), Naïve Bayes (NB), and Random Forest (RF) on the dataset they get and test it on Waikato environment for knowledge analysis. They used a dataset called CICAndMal2017 that have 10854 data (6500 generous and 4354 malware), and the dataset grouped into four noteworthy classifications (Adware, Ransomware, Scareware, SMS Malware). They show in this research that they detect malware location in 9 movements to accelerate the productivity of the activity classifiers. In addition, to characterize malware families, the model employs collection methodologies such as stream-based, bundle-based, and time-based highlights. The assessment exhibits the proposed incorporate set has more than 94 significant for real malware acknowledgment frameworks.

Utama et al. [18], proposed an algorithm for examining and identifying threat levels in android apps. The researchers used Naive Bayes (NB) algorithm to detect if the android application is dangerous or not. The dataset that they used contains 188,389 data. They analyzed this research on permissions and vulnerabilities. Furthermore, from this technique, they can distinguish that the android application is safe or not. Lastly, they get this research accuracy is 97.2%.

Yan et al. [19], investigate a system that detects malicious android applications using machine learning. The researchers got two datasets which are Malgenome and virus share. Also, they downloaded over 1000 applications from Google Play to test these applications. They solve the problem by collecting runtime logs from each application by utilizing the logs' data extricated. The researchers got a result with a false-positive rate below 8% and a true positive rate over 90%.

Rashidi et al. [20], proposed an algorithm that incorporates support vector machine (SVM) and active learning to identify malicious Android apps. The authors captured a log of applications using their custom instrumentation system, DroidCat. Following the collection of logs, they implemented scanning and parsing techniques. They used a dataset called Drebin Project. The dataset has more than 5000 applications from 179 malware families; they select 500 applications from malicious applications and 500 from benign applications to make training on it using RBF (Radial basis function) Kernel and 200 malicious applications and 200 benign applications to make tests on them through the researchers' model.

The running time per application to be 2-5 minutes that took 79 hours for all applications. Their exploratory outcomes exhibit that their proposed model accomplishes fulfilling true positive and false-positive rates and adjusts the recognition model for new malware patterns. Also, they used the Optimal Query Strategy (QQS), which gets a high impact on the model performance and accuracy.

Yuxia et al. [21], proposed a system that detecting malware android applications based on extreme learning machine. The researchers collected a dataset from the Tencent YingYongBao store. They used 524 benign applications and 525 malicious applications to test on them. To distinguish the android malicious application, they get permissions and API calls of the application using extreme learning machines. The results show their work excels with the current ones with minimal human intervention, better detection efficiency, and less detection time.

Lageman et al. [22], exploited a vulnerability in a device that can identify malicious Android apps based on their runtime actions. The researchers used logs and trace outputs to produce runtime datasets. They got the dataset from North Carolina State University's android malware Genome project. They tested the dataset using both Random Forest (RF) classifier and the support vector machine (SVM). They concluded that the Random Forest classifier outperforms the support vector machine in terms of true positive rate that exceeded 90% and a false-positive rate less than 6%.

IV. Malware Analysis Approach

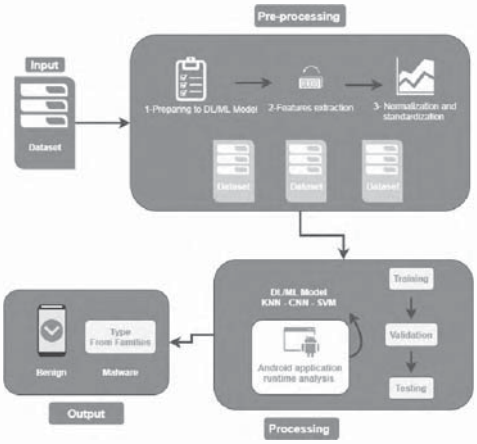

Fig. 3: Proposed Approach Work Flow

The development of the system is split into stages as in fig.3. Our malware detection problem is an example of a classification problem in machine learning. The input stage, in which the data is priorly collected From **Malgenome** data-set, contains 3800 unique Android applications. It also has permissions and API calls for benign and malicious applications and **Maldroid** data-set containing the benign applications, adware, banking malware, and mobile risk ware applications.

In the pre-processing phase, feature extraction is performed on our applications, getting the most common API calls and permissions from them to reduce the dimensionality of the raw data for our model to require less computation power and preparation of the proper features to be ready for training and testing. Normalization, standardization, and Anomaly detection will also be applied to the data-set if needed.

Our data-set is split into three subsets training, validation, and testing in the processing stage, respectively. We focus on training the different models using the training set; our malware detection model will be trained and tested on the data-set by some algorithms such as K-nearest neighbor, Naive Bayes, Support vector machine, and Decision tree by using dynamic analysis where we track the system API calls and permissions during the execution of an application which is executed during the applications' run-time and record the

results in a log file, Then we apply machine learning to differentiate between benign from malicious applications.

In the last stage, after scanning all applications, we will classify each selected application as either benign or which of the nine malware families (virus, worms, ransomware, spyware, etc.) these applications belong to according to their permissions and API calls and detecting it's dynamic behavior.

A. Data description

The data-sets used in building our model are called **Malgenome** data-set [23] and **Maldroid** [24]. Malgenome data-set consists of 1260 malware applications and 2539 benign; it has 3800 applications with a set of Permissions and API calls on each application and presents the most common permissions and API calls and their number of occurrence on malware and benign. Maldroid data-set contains benign applications, adware, banking malware, and mobile risk ware applications, which has 11599 applications.

V. Experiments

Our experiments are conducted on **Malgenome** and **Maldroid** data-sets. We implemented 4 different machine learning models, K-nearest neighbor, Naive Bayes, Support vector machine with linear kernel, and Decision tree Classifiers implemented by Python language. We excluded naive Bayes from Maldroid data-set because it has a very low accuracy of 51%. We use these classifiers due to their better accuracy and precision, especially the linear kernel with SVM algorithm with accuracy 0.86% on the Maldroid data-set and 99.9% on Malgenom. Data-sets are split into three subsets, training set, testing set, and validation, respectively, for the two data-sets.

Initially, with Malgenom: training, testing and validation applied on API calls and Permissions of applications. The first training set is fed to the models to adjust their parameters, and then we test 20% of the rest of the model to classify which app is benign or malware. Secondly, with Maldroid: training, testing, and validation applied on API calls to classify benign application, adware, banking malware, and mobile risk ware applications. Then the pre-trained model is tested using the testing set, achieving accuracy. We had a high accuracy with the Malgenom data-set and intermediate one with Maldroid as we excluded from it the naive Bayes algorithm. The tables I, II, and III illustrate the accuracy of the four classifiers on both data-sets.

The fig.4, 5, 6, 7, below depicts the common permissions and API calls with their number of occurrence in the malicious plotted in red and the benign plotted in blue with Malgenome data-set.

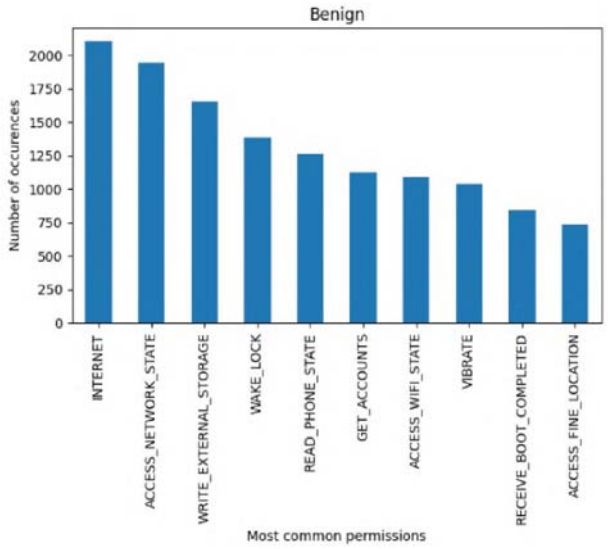

Fig. 4: A. Most Common Permissions with benign

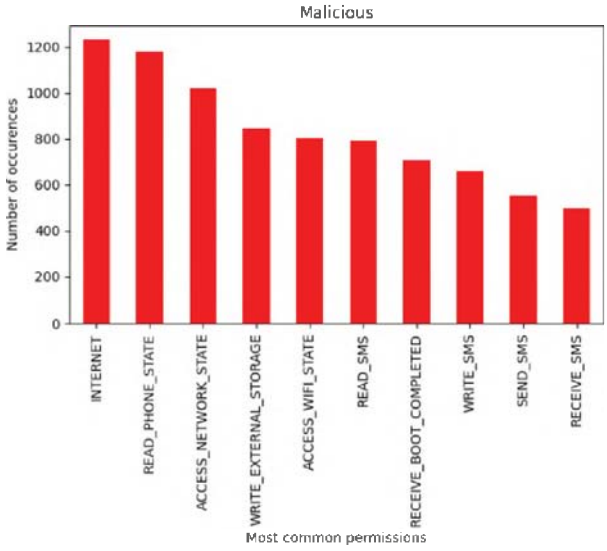

Fig. 5: B. Most Common permissions with Malware

TABLE I: API-Based accuracy results over Malgenom Dataset

Classifier	Accuracy	Precision		Recall		f1-score	
		Malware	Benign	Malware	Benign	Malware	Benign
KNN	0.97	0.98	1.00	1.00	1.00	1.00	1.00
Naive Bayes	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SVM	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Decision Tree	1.00	1.00	1.00	1.00	1.00	1.00	1.00

TABLE II: Permission-based Accuracy results over Malgenom Dataset

Classifier	Accuracy	Precision		Recall		f1-score	
		Malware	Benign	Malware	Benign	Malware	Benign
KNN	0.99	0.98	1.00	1.00	1.00	1.00	1.00
Naive Bayes	0.97	0.99	1.00	1.00	1.00	1.00	1.00
SVM	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Decision Tree	1.00	1.00	1.00	1.00	1.00	1.00	1.00

TABLE III: Accuracy results of API Calls over Maldroid Dataset

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	0.85	0.74	0.71	0.85	0.98	0.83	0.75	0.69	0.88	0.93	0.89	0.75	0.70	0.87	0.95	0.86
SVM	0.86	0.98	1.0	0.62	0.96	0.77	0.98	0.99	0.91	0.74	0.96	0.98	1.0	0.74	0.84	0.85
Decision Tree	0.88	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.62	1.0	1.0	1.0	0.0	0.76	1.0	1.0

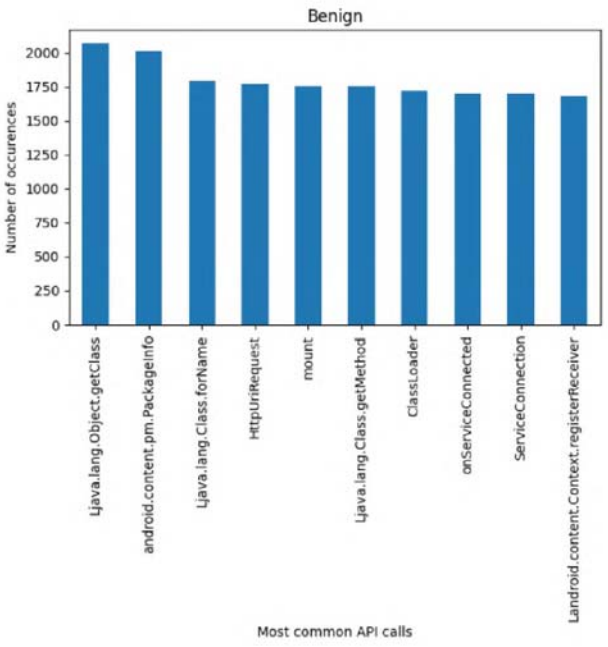

Fig. 6: A. Most Common API Calls with Benign

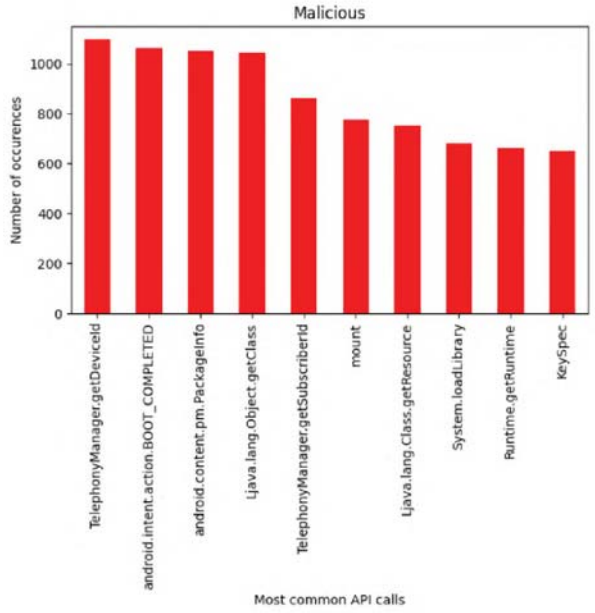

Fig. 7: B. Most Common API Calls with Malware

VI. Data Description

The data-sets used in building our model are called **Malgenome** data-set [23] and **Maldroid** [24]. Malgenome data-set consists of 1260 malware applications and 2539 benign, It has a total of 3800 applications with set of Permissions and API calls on each application and presenting the most common permissions and API calls and its number of occurrence on malware and benign. Maldroid data-set that contains benign application, adware, banking malware and mobile risk ware applications which has a total of 11599 applications.

VII. Conclusions and Future Work

We performed a precise feature selection over two data-sets. Still, we got a very high accuracy with Malgenom data-set with an average of 99% and less one with Maldroid with 86%, these results according to the accuracy average of 4 classifiers we performed the k-nearest neighbor, support vector machine, naive Bayes and decision tree due to their high accuracy and less processing time, We left a more sizeable feature selection exploration for future work with larger data-

set. This paper proposed a system for detecting malicious Android applications by tracking their dynamic behavior and based on their API calls and permissions. We will produce a mobile application combined with our machine learning model that scans all the apps to detect malware and benign ones in the future.

REFERENCES

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

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

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

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

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