International Journal of Social Sciences and Scientific Studies (2022)

Listes de contenus disponibles sur: Scholar

Enhancing Android Security: Analyzing Feature Sets in CICAndMal2017 and DroidFussion Datasets

Journal homepage: ijssass.com/index.php/ijssass

Enhancing Android Security: Analyzing Feature Sets in CICAndMal2017 and DroidFussion Datasets☆

Joshua Chibuike Sopuru [a,*]

[a] *Girne American University*

Received 6 January 2022; Accepted 23 May 2022
Available online 5 June 2022

ARTICLE INFO

Keywords:

Android dataset
Android Malware
Machine learning
Network-flow features

ABSTRACT

Because of the popularity of Android devices, many attackers spend lots of time and resources creating malicious applications aimed at breaching the security of Android device. Researchers on the other hand have not relented in seeking better ways of curbing attacks on Android devices. In other to achieve an efficient solution, researchers need large datasets to evaluate their solutions. Generating relevant data for this cause is however not an easy task, for this reason, several researchers rely on existing datasets.

In this paper, we evaluated the relevance of the feature sets of found in the CICAndMal2017 and DroidFussion datasets. During our study, we discovered the DroidFussion dataset has a higher variance and proved positive on some other parameters tested and as a result performed better. Results from the Random Forest classifier indicates that the Droid dataset achieved 90.0% precisions while the CICAndMal2017 achieved as low as 63% precision when tested following same conditions.

1. INTRODUCTION

From the time Android OS was introduced up until now, the use of Android OS among users of mobile devices has been on a steady rise. It is noticeable that one of the reasons behind this growth is the flexibility Android OS offers to both users and application developers. The availability of varieties of free mobile applications on Google Play store, the ease of installation coupled with a user-friendly operation environment is of no doubt a plus to Android OS. Despite the numerous advantages of Android OS there still exist some obvious security challenges posing risks to end-users. Researchers in this field have worked tirelessly on curtailing the effects of Malware on Android OS. In several works, Dataset gotten from other studies are used to evaluate several analysers. Before such a dataset can be used to evaluate classifiers, the dataset must meet certain requirements of which we intend to ravel in this article. In the Dataset section of this paper, we Compare datasets available to researchers and how they influence results gotten from their studies. Dataset plays a very important role in achieving an efficient analyser, however, the dataset on its own is not the only requirement towards classification efficiency. Combining the right machine classifier with the right dataset would yield a better result.

Our contribution is Thus as follows:

- Evaluating results gotten from different datasets used in training different algorithms.
- Defining requirements/specifications of the dataset for effective Android malware detection and categorization.
- Improving our existing Android malware detection model.

The subsequent sections of this paper are as follows: Section II presents a review of different works on Android Malware detection based on API calls and intent; Section III presents existing dataset available for researchers and a comparison of results gotten from different researches; Section IV presents our analysis framework and improvements are done on it; Section V concludes our research by presenting a summary of what we have done and the results gotten from our experiments.

2. LITERATURE REVIEW

Aung et al [1] did work using a machine learning-based malware detection system that identifies malware from benign and observes numerous grant characteristics. Further, to differentiate standard apps and malware apps based on the given sets, they used Random Forest and Decision tree algorithms.

However, to gather the malware, they used the K-means clustering algorithm. Their tests on two separate datasets result in a 90% detection rate on the average.

Additionally, research was made by Huang et al. [2]. In this study, they proposed the identification of more than 81% of different malicious samples using a permission-based mechanism that can also be used as a smart filter. They claimed that the experiment demands additional features to build a report that detects malicious applications.

More advanced approaches were taken to combine various samples in the detection of malware in apps, one example Droidmat [3] is analysing the behaviour of android apps. In this study, a malware analyser was used to analyse static data which includes API calls, intents, permissions and components for deployment. To improve the ability of the malware and, K-nearest Neighbour and K-means algorithm was applied to match the Apps malicious.

An early attempt was made by Nishimoto et al. [4] to use API call characteristics to recognize malware. A method was offered to acquire various phone IDs with the use of Logcat. While documenting the invocations, they inserted a Log.v scheme into the API calls. The experiment shows the possibility of detecting an application that is not capable of detection by static analysis.

After the experiment done by Nishimoto, research was carried out by Chan and Song [5]. They proposed a feature set that practically exceeded the performance of permissions feature set with the collection of specifically 19 API calls. The presentation contained API and permission set that aims at detecting malware. They claimed that the reports gotten from the experiments added value in identifying malware in Android.

In recent times, more experiment was carried out by researchers who contributed to the detection of malware using API sequence, API packages, and API dependency graphs. Following was the introduction of DroidSIFT[6] which differentiates malware using the API dependency graph. For Apps that were developed, security-related API graphs were stored by DroidSIFT in the database. critical. Efficient search of graphs in the database makes it possible to group Android packages that are similar and further, index them according to their identified risks in the APIs. As an example, an app was used as an experiment with this research, it brought together the index from the database and its critical APIs. The next phase was the extraction of vectors associated with the corresponding element that is in the graph database. If a non-zero vector is found amongst elements, it means that there are similarities of representation that are generated when comparing the database graph and the App graph. With Genome [7] dataset, 93% accuracy was reached when vector set was used to train signature-based classifies and anomaly detection.

In 2017 [8], research was based on the intents of Android to identify Malicious Application. It showed that malicious intents can be disclosed using Android intents when certain features like permissions are used. They also stated that intents are not enough in identifying malicious intents. Additional features need to be added. After carrying out experiments, results showed a 91% detection rate when Android intents are used corresponding to 83% of Android permissions. However, 96% was derived when both features were combined.

Another research R-PaackDroid [9] showcased that by the extraction of data from API packages, ransomware can be detected. The report gotten from the research showed that R-PackDroid accurately gathers applications without prior knowledge of the application strings of the ransomware.

Later in 2018,[10] some researchers were inspired to develop a methodology where dynamic and static analysing features which include API calls, privacy usage, permissions, and function call graphs can be used to detect android malware. Further, tools were designed to extricate corresponding characteristics such as system call, com+, and privacy leaking data. The combination of the proposed technique has the probability of detecting malware accurately of about 90% on three platforms named Bayes, KNN, and SVM.

From the time Android OS was introduced up until now, the use of Android OS among users of mobile devices has been on a steady rise. It is noticeable that one of the reasons behind this growth is the flexibility Android OS offers to both users and application developers. The availability of varieties of free mobile applications on Google Play store, the ease of installation coupled with a user-friendly operation environment is of no doubt a plus to Android OS. Despite the numerous advantages of Android OS there still exist some obvious security challenges posing risks to end-users. Researchers in this field have worked tirelessly on curtailing the effects of Malware on Android OS. In several works, Dataset gotten from other studies are used to evaluate several analysers. Before such a dataset can be used to evaluate classifiers, the dataset must meet certain requirements of which we intend to ravel in this article. In the Dataset section of this paper, we Compare datasets available to researchers and how they influence results gotten from their studies. Dataset plays a very important role in achieving an efficient analyser, however, the dataset on its own is not the only requirement towards classification efficiency. Combining the right machine classifier with the right dataset would yield a better result.

Our contribution is Thus as follows:

- Evaluating results gotten from different datasets used in training different algorithms.
- Defining requirements/specifications of the dataset for effective Android malware detection and categorization.
- Improving our existing Android malware detection model.

The subsequent sections of this paper are as follows: Section II presents a review of different works on Android Malware detection based on API calls and intent; Section III presents existing dataset available for researchers and a comparison of results gotten from different researches; Section IV presents our analysis framework and improvements are done on it; Section V concludes our research by presenting a summary of what we have done and the results gotten from our experiments.

I. LITERATURE REVIEW

Aung et al [1] did work using a machine learning-based malware detection system that identifies malware from benign and observes numerous grant characteristics. Further, to differentiate standard apps and malware apps based on the given sets, they used Random Forest and Decision tree algorithms. However, to gather the malware, they used the K-means clustering algorithm. Their tests on two separate datasets result in a 90% detection rate on the average.

Additionally, research was made by Huang et al. [2]. In this study, they proposed the identification of more than 81% of different malicious samples using a permission-based mechanism that can also be used as a smart filter. They claimed that the experiment demands additional features to build a report that detects malicious applications.

More advanced approaches were taken to combine various samples in the detection of malware in apps, one example Droidmat [3] is analysing the behaviour of android apps. In this study, a malware analyser was used to analyse static data which includes API calls, intents, permissions and components for deployment. To improve the ability of the malware and, K-nearest Neighbour and K-means algorithm was applied to match the Apps malicious.

An early attempt was made by Nishimoto et al. [4] to use API call characteristics to recognize malware. A method was offered to acquire various phone IDs with the use of Logcat. While documenting the invocations, they inserted a Log.v scheme into the API calls. The experiment shows the possibility of detecting an application that is not capable of detection by static analysis.

After the experiment done by Nishimoto, research was carried out by Chan and Song [5]. They proposed a feature set that practically exceeded the performance of permissions feature set with the collection of specifically 19 API calls. The presentation contained API and permission set that aims at detecting malware. They claimed that the reports gotten from the experiments added value in identifying malware in Android.

In recent times, more experiment was carried out by researchers who contributed to the detection of malware using API sequence, API packages, and API dependency graphs. Following was the introduction of DroidSIFT[6] which differentiates malware using the API dependency graph. For Apps that were developed, security-related API graphs were stored by DroidSIFT in the database. critical. Efficient search of graphs in the database makes it possible to group Android packages that are similar and further, index them according to their identified risks in the APIs. As an example, an app was used as an experiment with this research, it brought together the index from the database and its critical APIs. The next phase was the extraction of vectors associated with the corresponding element that is in the graph database. If a non-zero vector is found amongst elements, it means that there are similarities of representation that are generated when comparing the database graph and the App graph. With Genome [7] dataset, 93% accuracy was reached when vector set was used to train signature-based classifies and anomaly detection.

In 2017 [8], research was based on the intents of Android to identify Malicious Application. It showed that malicious intents can be disclosed using Android intents when certain features like permissions are used. They also stated that intents are not enough in identifying malicious intents. Additional features need to be added. After carrying out experiments, results showed a 91% detection rate when Android intents are used corresponding to 83% of Android permissions. However, 96% was derived when both features were combined.

Another research R-PaackDroid [9] showcased that by the extraction of data from API packages, ransomware can be detected. The report gotten from the research showed that R-PackDroid accurately gathers applications without prior knowledge of the application strings of the ransomware.

Later in 2018,[10] some researchers were inspired to develop a methodology where dynamic and static analysing features which include API calls, privacy usage, permissions, and function call graphs can be used to detect android malware. Further, tools were designed to extricate corresponding characteristics such as system call, com+, and privacy leaking data. The combination of the proposed technique has the probability of detecting malware accurately of about 90% on three platforms named Bayes, KNN, and SVM.

Captured Feature set found in CICAndMal2017 dataset

Src IP	Src Port	Dst IP	Dst Port	Protocol
Fwd Pkt Len Mean	Fwd Pkt Len Std	Bwd Pkt Len Max	Bwd Pkt Len Min	BwdPkt Len Mean
Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts
Bwd Pkt Len Std	Flow Byts/s	Flow Pkts/s	Flow IAT Mean	FlowIAT Std
TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Label	
Flow IAT Max	Flow IAT Min	Fwd IAT Tot	Fwd IAT Mean	
Fwd IAT Std	Fwd IAT Max	Fwd IAT Min	Bwd IAT Tot	Bwd IAT Mean
Bwd IAT Std	Bwd IAT Max	Bwd IAT Min	Fwd PSH Flags	Bwd PSH Flags
Fwd URG Flags	Bwd URG Flags	Fwd Header Len	Bwd Header Len	Fwd Pkts/s
Fwd Blk Rate Avg	Bwd Byts/b Avg	Bwd Pkts/b Avg	Bwd Blk Rate Avg	Subflow Fwd Pkts
Pkt Size Avg	Fwd Seg Size Avg	Bwd Seg Size Avg	Fwd Byts/b Avg	Fwd Pkts/b Avg
ACK Flag Cnt	URG Flag Cnt	CWE Flag Count	ECE Flag Cnt	Down/Up Ratio
Pkt Len Var	FIN Flag Cnt	SYN Flag Cnt	RST Flag Cnt	PSH Flag Cnt
Bwd Pkts/s	Pkt Len Min	Pkt Len Max	Pkt Len Mean	Pkt Len Std
Active Min	Idle Mean	Idle Std	Idle Max	Idle Min
Fwd Act Data Pkts	Fwd Seg Size Min	Active Mean	Active Std	Active Max
Subflow Fwd Byts	Subflow Bwd Pkts	Subflow Bwd Byts	Init Fwd Win Byts	Init Bwd Win Byts

different metrics are used for such selection, one of which is the

II. FEATURE SELECTION

There are different types of feature selection algorithm that can be applied on a dataset. These algorithms are classified into three main categories.

Filter based: This form of feature selection utilizes the predictive powers of features to make selections. Several

Pearson correlation. In a Pearson correlation, a value is returned indicating the significance of influence the features have. This is done by calculating the covariance between variables divided by their standard deviation.

Pearson correlation coefficient r can be computed with the

formula

$$r = \frac{1}{n-1} \sum \frac{(x_i - \overline{X})(y_i - \overline{Y})}{s_x s_y}$$

Where the values of r range from +1 to -1. R = 0 indicates no relationship exists.

Wrapper-based feature selection: Feature selection here is based on a selected learning algorithm. All possible combinations of the individual features are evaluated with respect to the set criterion. After evaluation, the sets of features with the optimal outcome is selected for that specific machine learning.

Embedded Feature selection: In this method, the algorithm for feature selection is embedded into the learning algorithm. A recursive feature elimination algorithm with Random Forest classifier (RFECV) is used on both datasets.

We evaluate two of our datasets with these feature selection methods and examine the results we get from each dataset and how they relate.

III. EXPERIMENT

Scene 1: Feature selection based on different algorithms:

Dataset: DroidFussion

Using the sklearn library found in python, we ran Chi-Squared tests on the dataset used for the development of DroidFussion.

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

Where O represents the observed outcomes in the dataset and E is the computed expected outcome gotten from our data. The Sklearn.feature_selection library automatically implements this formula on the dataset. Observations gotten from the DroidFussion dataset were recorded and we proceed to run some experiments on the CICAndMal2017 dataset.

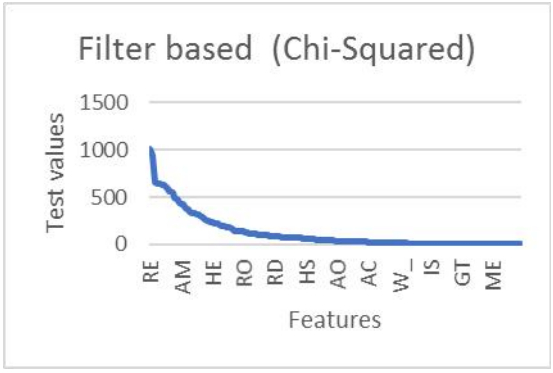

Fig 3: DroidFussion dataset Filter based feature selection

Distribution of feature relevance to model development (DroidFussion)

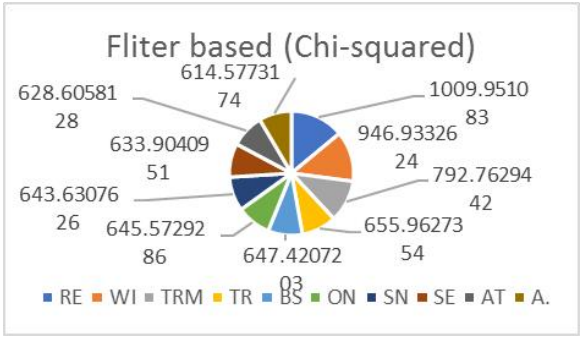

Fig 4: Chi-squared value of individual features

The percentage share of top 10 features found in the DroidFussion dataset relevant for machine learning (Chi-square)

We then evaluated the same dataset using an unsupervised learning technique (Variance Threshold). Here we are interested in eliminating features whose variance does not meet a certain threshold, fit it to a Random Forest Classifier (RFC) and compare its precision with features gotten selected using Chi-Squared.

Finally, we investigated Recursive Feature Elimination (RFE) on both datasets using RFC. results gotten showed a consistency in the rankings of selected features observed during the Chi-squared test.

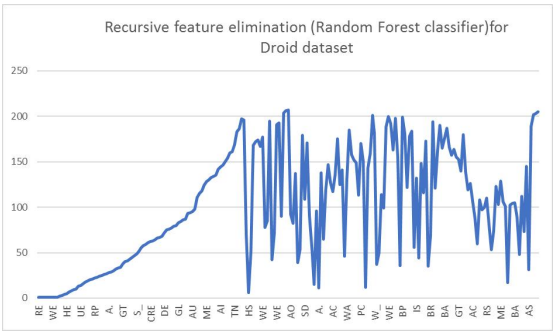

Fig 5: Droid dataset Recursive Feature Elimination

Scene 2: Feature selection based on different algorithms:

Dataset: CICAndMal2017

Distribution from Chi-Squared test conducted on the CICAndMal2017 dataset presented the result below:

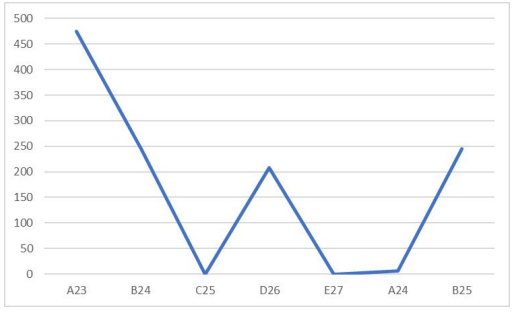

Fig 6: Distribution of feature relevance to model development (CICAndMal2017)

The percentage share of top 10 features found in the CICAndMal2017 dataset relevant for machine learning (Chi-square)

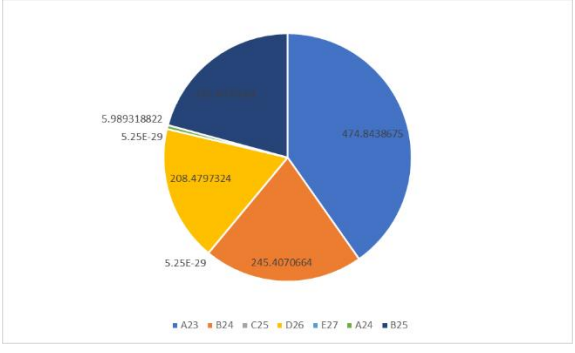

Fig 7: Chi-squared value of individual features

Finally, we investigated Recursive Feature Elimination (RFE) for the CICAndMal2017 dataset using Random Forest classifier as we did on the Droid dataset

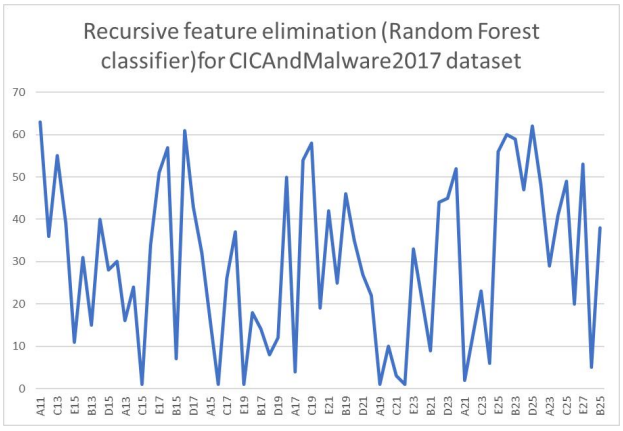

Fig 8: CICAndMal2017 dataset Recursive Feature Elimination

vi. RESULT

Our result presents a comparison of the two datasets based on the evaluated properties

Spread in features

For effective machine learning, the greater the differences of the features used, the more efficient the corresponding model will be. Hear we measure the spread in the features used in the Droid dataset and those used in the CICAndMal2017 dataset.

Computing the variance and standard deviation of each dataset in respect to the different experiments performed, we observed the following:

$$Deviation = X - \overline{X}$$

Variance

$$\sigma^2 = \frac{\sum (X - \overline{X})^2}{N}$$

Standard deviation

$$s^2 = \frac{\sum (X - \overline{X})^2}{N - 1}$$

$$s = \sqrt{s^2} = \sqrt{\frac{\sum (X - \overline{X})^2}{N - 1}}$$

Features listed in the CICAndMal2017 Dataset has a mean of 30.149, standard deviation of 19.24 and a variance of 370.40 While the Droid dataset had a mean of = 100.16, standard deviation of 61.93 and a variance of 3835.71.

The variance for the Droid dataset over a given mean (30.149) is 7.9228 and that of CICAndMal2017 over same mean is 0.9591. This shows that the features listed in Droid dataset is more suitable for classification when compared to those used in the CICAndMal2017 dataset.

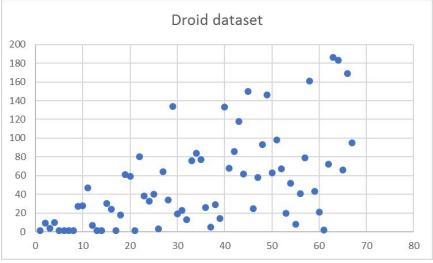

Fig 9: Spread in features represented in the Droid dataset

Results gotten from two machine learning classifier: Using Random forest classifier, Droid dataset achieved a precision of 0.906, 60% of the total sample fell under true positive, 5% false positive, 4% false negatives and 30% true negative. This shows that more than 90% of the samples were correctly classified.

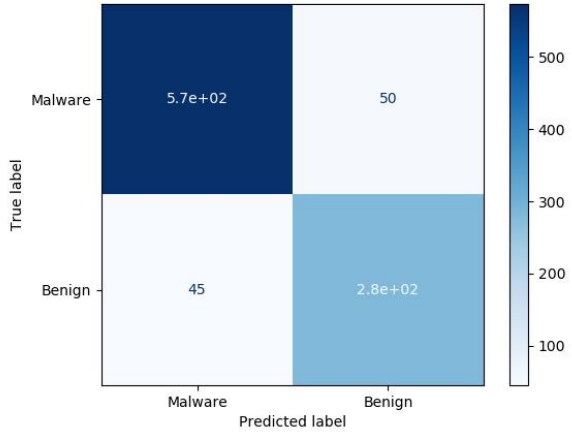

Fig 10: Confusion matrix for Droid dataset tested on RFC

On the other hand, CICAndMal2017 dataset, tested on a Random forest classifier achieved a precision of 0.63, 50.67% of the total sample fell under true positive, 9.28% false positive, and 26.68% false negative. This shows that 64.02% of the samples were correctly classified.

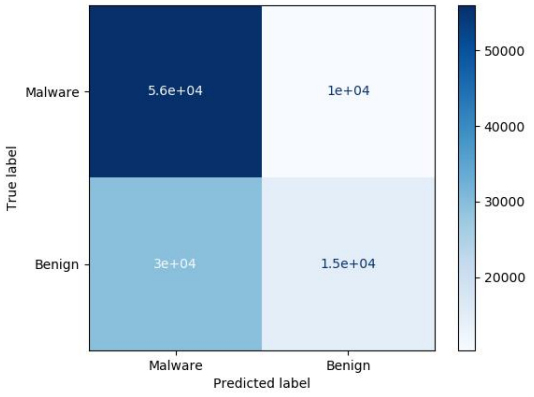

Fig 11: Confusion matrix for CICAndMal2017 dataset tested on RFC

vii. CONCLUSION

The need for Android malware detection and categorization has been a major concern to researchers. Using available datasets researchers try to evaluate the efficiency of different detection and categorization algorithms. In this research, we compared two main Malware datasets used in the past by researchers in developing malware detection engines. We evaluated variance and how it effects results on a Random forest classifier (RFC). Here we observed that in comparison to the Droid dataset, the CICAndMal2017 dataset performed poorly in several parameters tested. Using Chi-square and Recursive feature elimination methods, we selected seven features from each dataset and applied the RFC classifier on it. Results show that CICAndMal2017 dataset achieved a 0.63% accuracy on seven features while the Droid dataset achieved a 0.90% accuracy. This shows how important identification and collection of features with high variance influence learning.

Subsequently, we plan to revaluate the two datasets, create a subset of the intersection of features with same likelihood and develop a Bayesian network of their possible effect on classification.

REFERENCE

1- Z. Aung and W. Zaw. Permission-based android malware detection. International Journal of Scientific & Technology Research, 2(3):228–234, 2013.

2- C.-Y. Huang, Y.-T. Tsai, and C.-H. Hsu. Performance evaluation on permission-based detection for android malware.

In Advances in Intelligent Systems and Applications-Volume 2, pages 111–120. Springer, 2013

3- D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.- P. Wu. Droidmat: Android malware detection through manifest and api calls tracing. In 2012 Seventh Asia

Joint Conference on Information Security, pages 62–69. IEEE, 2012.

4- Y. Nishimoto, N. Kajiwara, S. Matsumoto, Y. Hori, and K. Sakurai. Detection of android api call using logging mechanism within android framework. In International Conference on Security and Privacy in Communication Systems, pages 393–404. Springer, 2013. 5- P. P. Chan and W.-K. Song. Static detection of android malware by using permissions and api calls. In 2014 International Conference on Machine Learning and Cybernetics, volume 1, pages 82–87. IEEE, 2014.

6- M. Zhang, Y. Duan, H. Yin, and Z. Zhao. Semanticsaware android malware classification using weighted contextual api dependency graphs. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pages 1105–1116. ACM, 2014.

7- Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In Security and Privacy (SP), 2012 IEEE Symposium on, pages 95–109. IEEE, 2012.

8- A. Feizollah, N. B. Anuar, R. Salleh, G. Suarez-Tangil, and S. Furnell. Androdialysis: Analysis of android intent effectiveness in malware detection. computers & security, 65:121–134, 2017.

9- D. Maiorca, F. Mercaldo, G. Giacinto, C. A. Visaggio,and F. Martinelli. R-packdroid: Api package-based characterization and detection of mobile ransomware.In Proceedings of the symposium on applied computing, pages 1718–1723. ACM, 2017.

10 -A. Desnos and P. Lantz. Droidbox: An android application sandbox for dynamic analysis. Lund Univ., Lund, Sweden, Tech. Rep, 2011.

Received 6 January 2022; Accepted 23 May 2022
Available online 5 June 2022