Significant API Calls in Android Malware Detection

Using Feature Selection Techniques and Correlation Based Feature Elimination

Asadullah Hill Galib
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh
bsse0712@iit.du.ac.bd

B M Mainul Hossain
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh
mainul@iit.du.ac.bd

Abstract— **Android API Calls are an important factor in differentiating malware from benign applications. Due to the increasing number of API Calls and considering computational complexity, the number of API calls in Android malware detection should be assessed and reduced without affecting detection performance. This study tries to figure out a feature reduction approach for identifying significant API Calls in Android malware detection. It incrementally analyzes various feature selection techniques to find out the minimal feature set and the most suitable technique. Also, it incorporated a correlation-based feature elimination strategy for further reduction of API Calls. Experiments on two benchmark datasets show that the Recursive Feature Elimination with Random Forest Classifier causes the minimal number of API Calls. Evaluation results indicate that the reduced set of significant API Calls (SigAPI) will perform relatively close to the full set of features in terms of accuracy, accuracy, recall, f-1 performance, AUC, and execution time. It also compares the performance with the existing malware detection works and the SigAPI outperforms most of the work regarding malware detection rate. Furthermore, it reports the top significant API Calls in malware detection. Finally, this work suggests that reduced features set of significant API Calls would be useful in classifying Android malware effectively.**

Keywords- Significant API Calls, Android Malware Detection, Feature Selection

I. INTRODUCTION

Android API (Application Programming Interface) is a series of specifications and guidelines that programs can follow to communicate with each other. Using API Calls this communication happens. APIs are growing exponentially every year [1]. Due to the wide-ranging applicability of API Calls, they are commonly used for characterizing and separating malware from benign applications.

However, the Android operating system uses a large number of API Calls and the number continues to expand. So, handling this large number of API calls in malware detection for Android is challenging. This would overfit the classifier model or complicated the classification method by providing a large number of features set. It would be useful to boost this problem by reducing features of the API Calls using feature selection techniques.

This study dealt with examining API Calls for reducing the irrelevant ones without tampering significant API Calls. It presents an approach for significant API Calls identification. Primarily, it incrementally employed several feature selection techniques. It trialed with Mutual Information Gain, Univariate ROC-AUC scores, Recursive Feature Elimination (RFE) with Gradient Boosting Classifier, and Random Forest Classifier, SelectKBest using chi scoring function, SelectFromModel using Random Forest and Extra Trees classifiers for exploring the effect of incremental feature selection. Subsequently, according to the performance evaluation, it infers a minimal range of features for different techniques and determines the best selection technique. Further, by incorporating a correlation-based feature elimination strategy, it reduces the minimal range of feature sets. Finally, the selected features are evaluated on two benchmark datasets based on five performance metrics (accuracy, precision, recall, f-1 score, AUC), execution time, and comparison with existing works.

Results show that from all the API Calls, 15-25 API Calls are significant in malware detection according to RFE with Random Forest Classifier. Evaluation illustrates that using those significant API calls, the performance is close enough to the full API Calls set. For instance, using the top 25 significant API Calls derived from the feature selection technique, the performance metrics are as follows for a particular dataset: accuracy - 97.07%, precision - 97.41%, recall - 94.60%, f-1 score - 0.960, and AUC – 0.993. Likewise, as far as the execution time is concerned, the significant API Calls take fairly less time. In comparison with existing works, this work outperforms other studies while using only a few numbers of API Calls. Also, the top significant API Calls are reported.

There were no prior works on reducing or defining significant API calls. To the best of our knowledge, this is the first study on significant API Calls in Android malware detection. The main contributions of the study are as follow:

- It proposes and assesses a feature reduction approach for identifying significant API Calls in Android Malware Detection effectively.
- It's reduced significant API Calls performs notably with respect to the full features set in terms of accuracy, precision, recall, f-1 score, AUC, and execution time. Also, it outperforms most of the existing works.
- It provides the top significant API Calls list in Android malware detection.

DOI reference number: 10.18293/SEKE2020-143

The rest of the paper is organized as follows. Section II presents the significant API Calls identification approach in detail. Section III gives the evaluation of the approach. Section IV gives the limitation. Section V gives related works. Section VI concludes the paper and guides future work.

II. Significant API Calls Identification approach

The overall approach of significant API Calls identification consists of five steps. The overview of the approach is depicted in Fig. 1. The details of each step are as follow:

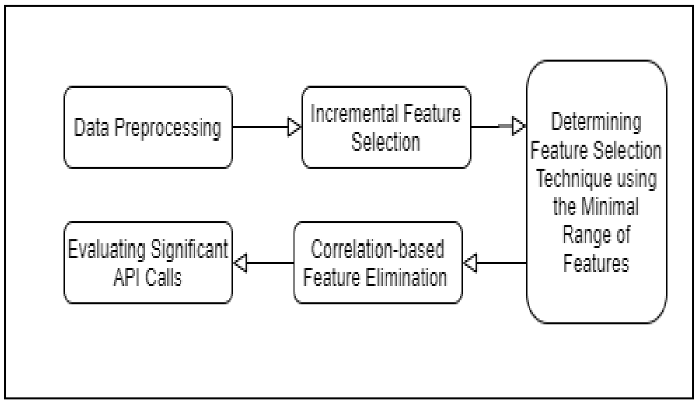
Data Preprocessing
Incremental Feature Selection
Determining Feature Selection Technique using the Minimal Range of Features
Evaluating Significant API Calls
Correlation-based Feature Elimination

Figure 1. *Significant API Calls Identification Approach*

A. *Data Preprocessing*

The dataset is preprocessed using traditional data preprocessing techniques. Other features are excluded from the dataset except for the API Calls. Subsequently, missing value treatment and label encoding are incorporated. For, incremental feature selection, the dataset is split into training and validation sets.

B. *Incremental Feature Selection (IFS)*

In identifying the significant API Calls, two aspects are considered. First, how many numbers of API Calls should we choose? In this regard, it is avoided to set any predefined parameters like a certain number of API Calls to be selected. Rather, it is intended to determine the optimal/minimal number of API Calls by analyzing performance metrics for different numbers of API Calls. In doing so, a feature selection technique is employed in an incremental way. For each feature selection technique, from one to the highest number of API Calls are assessed separately based on performance metrics.

Second, which feature selection technique is more suitable in reducing API Calls while maintaining the performance in Android malware detection? Again, various feature selection techniques are analyzed to figure out the most suitable technique, rather than imposing a predetermined feature selection technique. In this study, the following feature selection techniques are examined:

1. *Feature Selection using Mutual Information Gain (Entropy-Based):* Mutual Information is a non-negative value between two random variables, which measures dependency between variables. It measures the quantity of information gained by analyzing the other random variable involving one random variable. It is equal to zero if there are two independent random variables, and higher values mean higher dependence. The function is based on nonparametric methods based on entropy estimation of the distances from k-nearest neighbors as defined in [2] and [3].

2. *Feature Selection Based on Univariate ROC-AUC Score*: A ROC curve (receiver operating characteristic curve) is a graph representing a classification model output at all classification thresholds. This curve maps two parameters: True Positive Rate and False Positive Rate. AUC stands for "Area under the ROC Curve," meaning that AUC measures the whole two-dimensional space under the ROC Curve. The region under the curve (AUC) is proportional to the probability that a classifier ranks a randomly selected positive instance higher than a randomly selected negative one by using normalized units [4]. Univariate ROC-AUC involves the analysis of a single variable. An AUC equal to 0.5 corresponds to a type of random classification. For a model to be acceptable AUC will be greater than 0.5.

3. *Feature Selection using Recursive Feature Elimination (RFE):* The goal of recursive feature elimination (RFE) is to pick features by recursively considering smaller and smaller sets of features, given an external estimator that assigns weights to features. First, the estimator is trained on the initial collection of features and the importance of each function is obtained. The least significant characteristics are then pruned from the present range of characteristics. The process is repeated recursively on the pruned collection before finally achieving the required number of features to be chosen [5]. In this work, two classifiers are used as the estimators of the RFE.

   3.1. *RFE with Gradient Boosting Classifier*: As the base estimator of the RFE, Gradient Boosting Classifier is employed. Gradient Boosting Classifier builds an additive model in forward-stage-wise fashion; enables arbitrary differentiable loss functions to be optimized. Regression trees are fit on the negative gradient of the function of binomial or multinomial loss of deviance in each point [6].

   3.2. *RFE with Random Forest Classifier*: Random Forest Classifier is also used as the base estimator of the RFE. It is a meta-estimator that fits multiple decision tree classifiers on various dataset sub-samples and uses an average to improve predictive [7].

4. Feature Selection using SelectKBest with chi2: SelectKBest scores the features according to the k highest scores. It takes a score function as a parameter, which would be specific to a pair. The score function retains the features of the first k with the highest scores [8]. In this study, the chi2 scoring function is employed. This scoring function computes the chi-squared stats between each non-negative feature and class scores accordingly. It tests for which the distribution of the test statistic approaches the $\chi 2$ (Chi-Squared) distribution asymptotically [9].

5. Feature Selection using SelectFromModel (Tree-Based): SelectFromModel is a meta-transformer that can be used

along with any tree-based estimator. It calculates the feature importance of each feature according to fitting the estimator into the data. Based on the feature importance it selects the top N features, where N is predefined [10]. Tree-based estimators are used here as it can classify the significant features by selecting the classification features on the basis of how well they boost the node's purity [11]. In this case, every possible value of N is evaluated. Also, two tree-based estimators are incorporated here: Random Forest Classifier and Extra Trees Classifier.

C. *Determining Feature Selection Technique using the Minimal Range of Features*

After implementing the incremental feature selection using different feature selection techniques, analysis of performance metrics is carried out to identify the minimal range of features. The minimal range of features implies a range of features from which segment the performances of Android malware detection are not increased significantly with respect to the increase of features. In other words, before the minimal range, the performances are increased. But, after the minimal range, the performances are quite unchanged with the increase in the number of features. To draw a conclusion from the analysis, a self-explanatory plot is generated using the performance metrics (accuracy, precision, recall, f-1 score) with respect to the increasing number of features. According to the plots for different techniques, the minimal range of features are deduced.

These minimal ranges are conducive to determine the best feature selection technique. The feature selection technique with the lowest minimal range is carefully chosen for identifying significant API Calls in Android Malware Detection.

D. *Correlation-based Feature Elimination*

After selecting the important features using the suitable feature selection technique, a final feature elimination strategy is performed for further reduction of API Calls without affecting the performances notably. Here, a correlation-based feature elimination strategy is applied.

A pair-wise Pearson correlation coefficient is calculated for all pairs of important API Calls. It is a measure of the linear correlation between two variables X and Y. It is calculated using the following equation [12]:

$$\rho_{xy} = \frac{\mathrm{Cov(x,y)}}{\sigma_x\,\sigma_y} \qquad (1)$$

Where,

ρ_{xy} = Pearson correlation coefficient

Cov (x,y) = covariance of variable x and y

σ_x = standard deviation of x

σ_y = standard deviation of y

Then all the pairs with a Pearson correlation coefficient greater than 0.85 are filtered out for the feature elimination process. As the two features in each pair are highly correlated, so removing one of them would not affect the classification performances.

The elimination strategy here is to remove the less important feature from each pair. To measure the relative importance of the features, a tree-based estimator – Random Forest Classifier is used. According to the relative feature importance, the more important feature in each pair is intact, and the less important feature is eliminated.

E. *Evaluating Significant API Calls*

Finally, the reduced set of API Calls are evaluated according to five performance metrics – accuracy, precision, recall, f-1 score, AUC. Here, the minimal range of features derived from the best feature selection technique is assessed. In the final assessment, the Random Forest classifier is trained and evaluated with 10-fold cross-validation. Along with the five-performance metrics, execution time, and comparison with the existing approach are also evaluated. Also, significant API Calls are determined and reported.

III. Evaluation

In the evaluation of this study, two benchmark datasets are used. The experimental results are analyzed based on five performance metrics - accuracy, precision, recall, f-1 score, and AUC. These metrics are widely used in performance measure of Android malware detection. Besides, the execution time of detection is considered for evaluation. Three research questions are being answered here regarding significant API Calls in malware detection.

A. *Dataset*

In this study, the Drebin [13] and the Android Malware Genome Project [14] datasets are used. The datasets are used separately to ensure the applicability and generalizability of the approach.

The Drebin dataset contains 5,560 malware applications from 179 different malware families. Also, 9470 benign applications derived from the Google Play Store are incorporated here for classifying the malware properly.

The Android Malware Genome Project dataset contains 1,200 malware samples that cover most existing Android malware families. Here, 2539 benign applications derived from the Google Play Store are incorporated. In the rest of the paper, this dataset is referred to as Malgenome.

Only the API Calls are considered in this work. In total, 73 API Calls are found in the Drebin dataset and 69 API Calls are found in the Android Malware Genome Project dataset.

B. *RQ1: How can we sort out the significant API Calls (features) in Android malware detection?*

According to the different incremental feature selection techniques, the minimal ranges are analyzed to determine the best feature selection technique in identifying significant API Calls. The near minimal ranges are as depicted in Table I.

From the experiments, the best feature selection technique for API Calls reduction is Recursive Feature Elimination (RFE) with Random Forest Classifier as it has the lowest minimal range among other techniques for both datasets (see Table I).

TABLE I. MINIMAL RANGE OF FEATURES FOR DIFFERENT FEATURE SELECTION TECHNIQUES

Feature Selection Technique	Minimal Range for Drebin	Minimal Range for Malgenome
Mutual Information Gain	37-42	23-28
Univariate ROC-AUC Score	35-38	25-30
RFE with Gradient Boosting Classifier	23-28	20-25
RFE with Random Forest Classifier	18-25	18-21
SelectKBest with chi2	47-50	30-33
SelectFromModel with Random Forest Classifier	25-30	17-22
SelectFromModel with Extra Trees Classifier	28-33	20-23

The minimal range of features using RFE with Random Forest Classifier for the Drebin dataset can be deduced from Fig.2 and Fig. 3. According to Fig. 2., with the number of features, are increased, the performance metrics are also increased initially. However, in the range between 18-25 features (approximately), the performance metrics are going to be stable and remain constant (the lines are almost horizontal) for the following selected features. So, it can be inferred that by taking those 18-25 features, the performances are close enough to the actual performances of all the 73 features. In the next research question, this minimal range of features are evaluated.

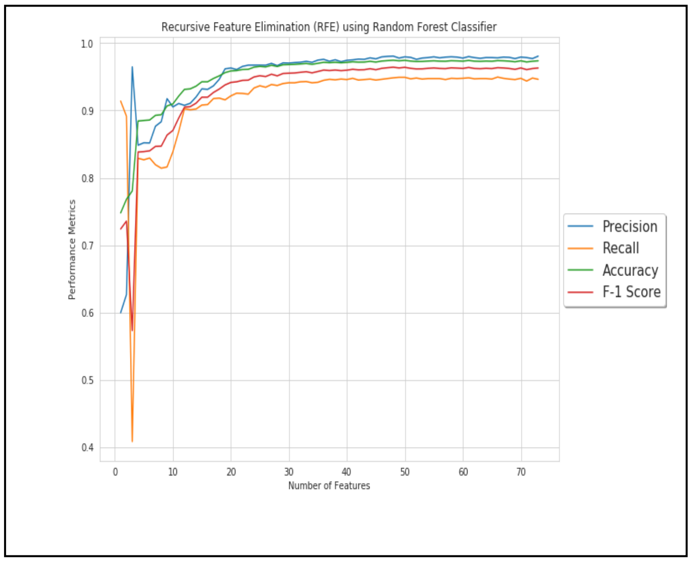

Figure 2. Recursive Feature Elimination (RFE) using Random Forest Classifier (Performance Metrics vs Number of Features)

Before proceeding to the performance evaluation, the correlation-based feature elimination (CFE) is performed on the minimal range of features. And, experiments show that CFE can reduce the features as depicted in Table II. For instance, CFE can reduce 18 features to 15 and 16 features respectively for the Drebin and Malgenome datasets.

TABLE II. CORRELATION-BASED FEATURE ELIMINATION (CFE) ON THE MINIMAL FEATURE SETS

Number of API Calls		
Minimal Feature Sets	With CFE for Drebin	With CFE for Malgenome
18	15	16
19	15	17
20	17	18
21	19	19
22	20	20
23	21	21
24	21	21
25	22	22

C. *RQ2: How do the significant API Calls perform in detecting Android Malware?*

In this research question, the reduced set of Significant API Calls (SigAPI) are evaluated based on the performance metrics, execution time, and comparison with existing works.

1) *Performance Evaluation of the SigAPI*

The performance evaluation for the significant API Calls is based on five metrics. The evaluation is described in Table III and Table IV for the number of API Calls– 15, 17, 19, 21, 23, 25, and all API Calls to sidestep redundancy.

Table III shows that for the Drebin dataset, the performance metrics using significant API Calls are close to the performance metrics of using all the API Calls (73).

Table IV also shows that for the Malgenome dataset, the significant API Calls performs almost identically to the full feature set of API Calls (69).

Specifically, how many significant API calls should be selected? - It depends on the requirement of the stakeholders. However, it is suggested to use the range of 15-25 significant API Calls based on the prerequisite of performance metrics.

TABLE III. PERFORMANCE EVALUATION OF THE SIGNIFICANT API CALLS (DREBIN)

Feature Selection Technique	# of API Calls	Acc (%)	Pre (%)	Rec (%)	F-1	AUC
All Features	73	98.32	98.62	96.17	0.974	0.996
RFE with Random Forest Classifier	25	97.07	97.41	94.60	0.960	0.993
	23	96.69	96.92	94.06	0.955	0.993
	21	96.26	96.64	93.15	0.949	0.992
	19	96.17	96.26	93.27	0.947	0.991
	17	95.62	95.58	92.43	0.940	0.988
	15	95.38	96.03	91.29	0.936	0.986

TABLE IV. PERFORMANCE EVALUATION OF THE SIGNIFICANT API CALLS (MALGENOME)

Feature Selection Technique	# of API Calls	Acc (%)	Pre (%)	Rec (%)	F-1	AUC
All Features	69	98.71	98.87	97.22	0.980	0.998
RFE with Random Forest Clas-sifier	25	98.12	98.16	96.19	0.972	0.998
	23	98.03	98.15	95.87	0.970	0.998
	21	98.10	98.07	96.10	0.971	0.996
	19	96.16	97.92	96.51	0.972	0.996
	17	97.97	97.82	96.03	0.969	0.995
	15	97.26	97.62	94.05	0.958	0.993

2) *Execution Time of the SigAPI*

Table V shows the comparative execution time of malware detection. The result shows that using the significant API Calls (ranges between 15-25), the execution time of the malware detection is considerably lower than using all the features. For large data sets, this time would be substantially higher.

TABLE V. EXECUTION TIME OF THE SIGNIFICANT API CALLS

# of API Calls	Execution Time (s) for Drebin	Execution Time (s) for Malgenome
All	6.48	5.76
25	4.15	4.02
23	4.11	3.98
21	4.11	3.88
19	3.95	3.85
17	3.91	3.78
15	3.90	3.74

3) *Comparison with Existing Works*

Table VI shows the comparative analysis of the detection rate using significant 20 API Calls – SigAPI (20) for the Drebin dataset with respect to some existing works on Android malware detection. The result shows that SigAPI (20) outperformed all the existing works except two.

TABLE VI. COMPARISON WITH THE EXISTING WORKS

Works	Detection Rate (%)
SigAPI (20)	96.30
Drebin [10]	93.90
SigPID [15]	93.62
Yerima et al. [11]	92.1%
Yerima et al. [12]	97.5%
Peiravian et al. [13]	95.75%
DroidAPIMiner [14]	~99%
Altaher et al. [16]	91%

A. *RQ3: Which API Calls are Significant in Android Malware Detection?*

Table VII shows the top 25 significant API Calls in Malware Detection for the Drebin dataset. These API Calls are derived from the feature selection technique – RFE with Random Forest Classifier. Also, these 25 API Calls are almost identical to the Malgenome dataset except 3 API Calls. More data instances would be conducive to generating identical API Calls. Yet, as this study primarily suggests a feature reduction approach for significant API Calls, the dataset to dataset it may slightly vary due to the inconsistency and time period of datasets.

TABLE VII. TOP 25 SIGNIFICANT API CALLS (DREBIN)

TOP 25 SIGNIFICANT API CALLS (DREBIN)	
transact	ClassLoader
onServiceConnected	Landroid.content.Context.registerReceiver
bindService	Ljava.lang.Class.getField
attachInterface	android.content.pm.PackageInfo
ServiceConnection	TelephonyManager.getLine1Number
android.os.Binder	Ljava.lang.Class.getMethod
Ljava.lang.Class.getCanonicalName	android.telephony.gsm.SmsManager
Ljava.lang.Class.getMethods	TelephonyManager.getSubscriberId
Ljava.lang.Class.cast	Ljava.lang.Object.getClass
Ljava.net.URLDecoder	TelephonyManager.getDeviceId
android.content.pm.Signature	HttpUriRequest
android.telephony.SmsManager	Runtime.exec

IV. LIMITATION

In this study, only the Drebin and Malgenome datasets have been analyzed, which is subject to bias and lack of generalizability, threatening external validity. In terms of threats to internal validity, the parameters of different techniques and algorithms, execution time measurement are susceptible to bias and can be examined differently by different analysts and machines.

V. RELATED WORK

Several works dealt with API Calls in Android malware detection. For instance, Drebin incorporated API Calls with other features and obtained an accuracy of 93.90% in malware detection [13]. Yerima et al. combined API Calls and Permissions with Bayesian Classifier and attained 92.1% accuracy [15]. In another work, they achieved an accuracy of 97.5% and an AUC of 0.953 by using a composite parallel classifier approach [16]. Using API Calls modeled with SVM (Support Vector Machine), Peiravian et al. gained an accuracy of 95.75% and an AUC of 0.957 [17]. Likewise, DroidAPIMiner integrated API level features and reached an accuracy as high as 99% using the KNN classifier [18].

Though a handful number of works employed API Calls, none dealt with reducing API Calls or identifying important API Calls. However, feature reduction technique is applied in Permission features previously. Li et al. successfully reduced 135 Permission features to 22 features. They used Permission ranking with negative rate, support based Permission ranking, and Permission mining with association rules for feature selection. Their reduced Permission features have higher recall value, close enough accuracy value with the full features set. But their precision was lower and false positive rate (FPR) was higher significantly with respect to all Permission features [19].

Altaher et al. proposed an approach based on ANFIS with fuzzy c-means clustering using significant application permissions. Their classification accuracy was 91%, with the lowest false positive and false negative rates of 0.5% and 0.4%, respectively [20].

Wang et al. evaluated individual permissions and collective permissions and implemented three measures of scoring on the permission features. They discovered dangerous permission subsets using Sequential Forward Selection (SFS) and Principal Component Analysis (PCA). They got a 94.62% detection rate [21].

To the best of our knowledge, there is no such work on feature reduction of API Calls in Android malware detection.

VI. Conclusion

In this study, a feature reduction approach is proposed for identifying significant API Calls in Android Malware detection. Evaluation of the significant API Calls shows that API Calls can be reduced without affecting performance so much. Therefore, reduced features set of significant API Calls would be convenient in classifying Android malware considering performance and computational complexity.

In the future, the approach will be evaluated using different and large datasets. Also, other feature selection techniques using deep learning, ensemble learning, etc. will be employed.

References

[1] J. Fernando, "What is an API ? How to call an API from Android ?," DroidMentor, 12-Oct-2016. [Online]. Available: https://droidmentor.com/api-call-api-android/. [Accessed: 11-Jan-2020].

[2] A. Kraskov, H. Stögbauer, and P. Grassberger, "Erratum: Estimating mutual information [Phys. Rev. E69, 066138 (2004)]," Physical Review E, vol. 83, no. 1, 2011.

[3] B. C. Ross, "Mutual Information between Discrete and Continuous Data Sets," PLoS ONE, vol. 9, no. 2, 2014.

[4] T. Fawcett, "An introduction to ROC analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006.

[5] "sklearn.feature_selection.RFE¶," scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html. [Accessed: 11-Feb-2020].

[6] M. B. Fraj, "In Depth: Parameter tuning for Gradient Boosting," Medium, 24-Dec-2017. [Online]. Available: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-gradient-boosting-3363992e9bae. [Accessed: 03-Mar-2020].

[7] Ho, Tin Kam. "Random decision forests." In Proceedings of 3rd international conference on document analysis and recognition, vol. 1, pp. 278-282. IEEE, 1995.

[8] "sklearn.feature_selection.SelectKBest¶," scikit. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html. [Accessed: 11-Jan-2020].

[9] Pearson K. X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science. 1900 Jul 1;50(302):157-75.

[10] "1.13. Feature selection¶," scikit. [Online]. Available: https://scikit-learn.org/stable/modules/feature_selection.html. [Accessed: 01-Mar-2020].

[11] "Using Scikit-Learn in python for feature selection," Data Science Beginners, 26-Nov-2018. [Online]. Available: https://datasciencebeginners.com/2018/11/26/using-scikit-learn-in-python-for-feature-selection/. [Accessed: 01-Mar-2020].

[12] "Basic Concepts of Correlation," Real Statistics Using Excel. [Online]. Available: http://www.real-statistics.com/correlation/basic-concepts-correlation/. [Accessed: 04-Mar-2020].

[13] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," Proceedings 2014 Network and Distributed System Security Symposium, 2014.

[14] Yajin Zhou, Xuxian Jiang, "Dissecting Android Malware: Characterization and Evolution," Proceedings of the 33rd IEEE Symposium on Security and Privacy (Oakland 2012), San Francisco, CA, May 2012

[15] Yerima, S. Y., Sezer, S., McWilliams, G., & Muttik, I. (2013, March). A new android malware detection approach using bayesian classification. In 2013 IEEE 27th international conference on advanced information networking and applications (AINA) (pp. 121-128). IEEE.

[16] Peiravian, N., & Zhu, X. (2013, November). Machine learning for android malware detection using permission and api calls. In 2013 IEEE 25th international conference on tools with artificial intelligence (pp. 300-305). IEEE.

[17] Yerima, S. Y., Sezer, S., & Muttik, I. (2014, September). Android malware detection using parallel machine learning classifiers. In 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies (pp. 37-42). IEEE.

[18] Aafer, Y., Du, W., & Yin, H. (2013, September). Droidapiminer: Mining api-level features for robust malware detection in android. In International conference on security and privacy in communication systems (pp. 86-103). Springer, Cham.

[19] Li, J., Sun, L., Yan, Q., Li, Z., Srisa-an, W., & Ye, H. (2018). Significant permission identification for machine-learning-based android malware detection. IEEE Transactions on Industrial Informatics, 14(7), 3216-3225.

[20] Altaher, A., & BaRukab, O. (2017). Android malware classification based on ANFIS with fuzzy c-means clustering using significant application permissions. Turkish Journal of Electrical Engineering & Computer Sciences, 25(3), 2232-2242.

[21] Wang, W., Wang, X., Feng, D., Liu, J., Han, Z., & Zhang, X. (2014). Exploring permission-induced risk in android applications for malicious application detection. IEEE Transactions on Information Forensics and Security, 9(11), 1869-1882