Assessing Important API Calls using Feature Selection Techniques in Android Malware Detection

Asadullah Hill Galib and B M Mainul Hossain

[1,2] University of Dhaka, Nilkhet Rd, Dhaka 1000, Bangladesh
lncs@springer.com

Abstract. Android API Calls play a vital role in differentiating malware from benign applications. Because of the increasing number of API Calls and considering the over-fitting problem and computation complexity, API Calls should be analyzed for filtering out irrelevant features. This study aims at figuring out important API Calls for malware detection. It utilizes various feature selection techniques in an incremental approach. It explores the effect on performance metrics by number of features. Analysis shows that feature selection using Recursive Feature Elimination with Random Forest Classifier generates the optimal number of features (20-25) form 73 API features. It also evaluates the reduced features set based on performance, execution time, and comparative analysis with the existing approaches. Results show that, the reduced features (26) outperformed existing works: accuracy - 97.08%, precision - 97.29%, recall - 94.75%, f-1 score - 0.960, and AUC – 0.993. Besides, this work presents an insight into the influence of incremental feature selection in performance analysis. Finally, the identification of the important API Calls is performed. This study suggests that reduced features set of API Calls would be beneficial in classifying Android malware.

Keywords: Important API Calls, Android Malware Detection, Feature Selection.

1 Introduction

The Android ecosystem offers an API interface that can be used by developers to communicate with the Android operating structure. The API interface is a central series of modules and classes. Due to the wide-ranging applicability of API Calls, they are commonly used for characterizing and separating malware from benign applications.

However, a large number of API Calls is used in the Android operating system and the number is expanding unceasingly. So, dealing with such large number API features in Android malware detection are problematic. This might overfit the classifier model or intricate the classification process by presenting a large number of features set. To alleviate this issue, feature reduction of API Calls features using feature selection techniques would be useful.

This work dealt with analyzing API Calls for reducing the irrelevant features without tampering important features. First, it utilized a variety of feature selection techniques in an incremental way. It experimented 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 for investigating the influence incremental feature selection. According to the incremental feature sets, an optimal range of features are selected. The selected features are further evaluated based on performance metrics (accuracy, precision, recall, f-1 score, AUC), execution time, and comparison with existing works. Also, the top important features are reported.

Results show that from the 73 API features, 20-25 features are important according to RFE with Random Forest Classifier. Evaluation depicts that using those important API calls, the performance is close enough with respect to the full feature set. For instance, using 26 most important features derived from the feature selection technique, the performance metrics are as follow: accuracy - 97.08%, precision - 97.29%, recall - 94.75%, f-1 score - 0.960, and AUC – 0.993. Also, in terms of execution time, the important features set takes quite less time. In comparison with existing works, this study outperforms most of the works while using only a few numbers of important API Calls.

No prior work dealt with feature reduction of API Calls or identification of critical API Calls. To the best of our knowledge, this is the first study on assessing feature selection in API Calls reduction for Android malware detection. This study tends to answer the following research questions:

- RQ1: What extent of API Calls are important in Android malware detection?
- RQ2: How do the important API Calls perform in detecting Android Malware?
- RQ3: Which API Calls are critical in Android Malware Detection?

2 Methodology

The methodology of this study consists of five steps. The overview of the methodology is depicted in Fig. 1.

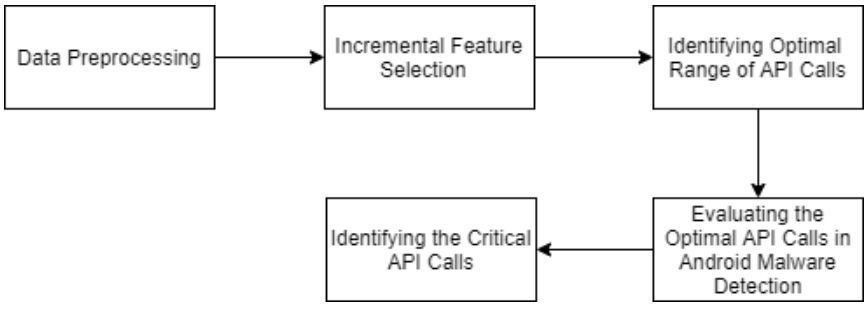

Fig. 1. Important API Calls Assessment for Android Malware Detection

2.1 Data Preprocessing

The dataset is preprocessed using the conventional data preprocessing techniques. First, all other features except the API Calls are removed from the dataset. Afterward missing value treatment and label encoding is employed. For, incremental feature selection, the dataset is split into training and testing sets.

2.2 Incremental Feature Selection

To analyze the feature importance and answer the first research question, various feature selection techniques are employed in an incremental fashion. In the evaluation of incremental feature selection, the Random Forest classifier is used. Here, incrementally all API Calls are evaluated to figure out the trends in performances and to identify an optimal number of features considering performances as well as computation complexity. The following feature selection techniques are used in this study:

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 [1] and [2]. The mutual information of two jointly discrete random variables X and Y is calculated as a double sum [3]:

$$I(X;Y) = \sum \sum_{p(X,Y)}(x,y) \log \left(\frac{p(X,Y)^{(x,y)}}{pX^{(x)}pY^{(y)}}\right) \quad (1)$$

where $p_{(X,Y)}$ is the joint probability mass function of X and Y, and pX and pY are the marginal probability mass functions of X and Y respectively.

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.

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 fea-

tures. First, the estimator is trained on the initial collection of features and the importance of each function is obtained by either a coef_ attribute or a feature_importances_ attribute. 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 estimator of the RFE.

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].

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 average to improve predictive performance and control over-fitting [7].

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].

2.3 Identifying Optimal Range of API Calls

For the different feature selection techniques, graphs are plotted on the performance metrics vs the number of features selected. Five performance metrics are considered in the malware classification problems: accuracy, precision, recall, f-1 score, AUC (ROC) score. According to the graph, the performance metrics are analyzed with respect to different numbers of features to figure out any discerning pattern in identifying the optimal number/range of features (API Calls).

2.4 Evaluating the Optimal API Calls in Android Malware Detection

After getting the optimal ranges of API Calls derived from different feature selection techniques, the optimal numbers of API Calls are evaluated in Android Malware Detection. Here, the optimal feature set derived from the best feature selection technique is assessed. Random Forest classifier with the optimal feature set 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.

2.5 Identifying the Critical API Calls

Finally, the critical API Calls are identified by utilizing the best feature selection technique. Experimentally, these API Calls are the top N number of critical API features in Android malware detection. These critical API Calls are based on the dataset used in this study.

3 Evaluation

3.1 Data set

In this study, the Drebin [10] dataset is used. The dataset contains 5,560 malware applications from 179 different malware families. Also, 9470 benign applications derived from the Drebin dataset are incorporated here for balancing the dataset.

Only the API Calls are considered in this work. In total, 73 API Calls are found in the dataset.

3.2 The Extent of Important API Calls in Android Malware Detection (RQ1)

In this section, the results from different feature selection techniques are analyzed to find out insightful trends in the performance metrics regarding the increasing number of features.

Feature Selection using Mutual Information Gain (Entropy-Based): In the incremental feature selection setup using Mutual Information Gain, the performance metrics, such as Accuracy, Precision, Recall, F-1 Score, ROC-AUC Score depict an intuitive pattern (see Fig. 2 and Fig. 3). According to Fig. 2, as the number of features increases, the performances of the model also increases (except for recall, a downward change in between 10 to 15 features). But, after selecting 36-38 features, the overall performances of the model remain almost constant as the lines proceed horizontally. Also, for ROC-AUC scores using different numbers of features, the pattern persists; after selecting 36-38 features, the line spreads horizontally (see Fig. 3).

This trend signifies that all the 73 API Calls are not indicative of Android malware detection. Mostly, 36-38 features (API Calls) are responsible for impacting the performances of malware detection.

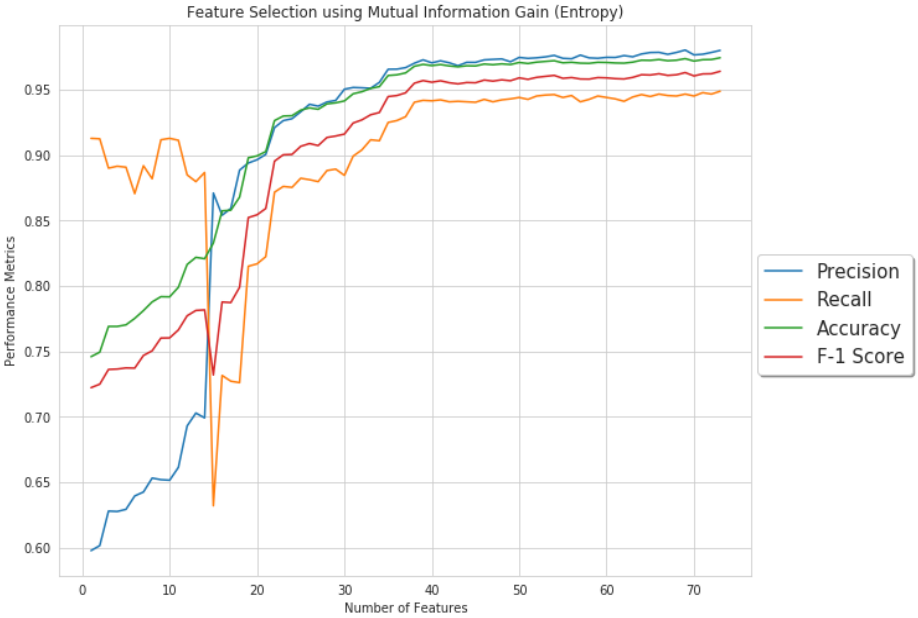

Fig. 2. Performance Metrics vs Number of Features using Mutual Information Gain

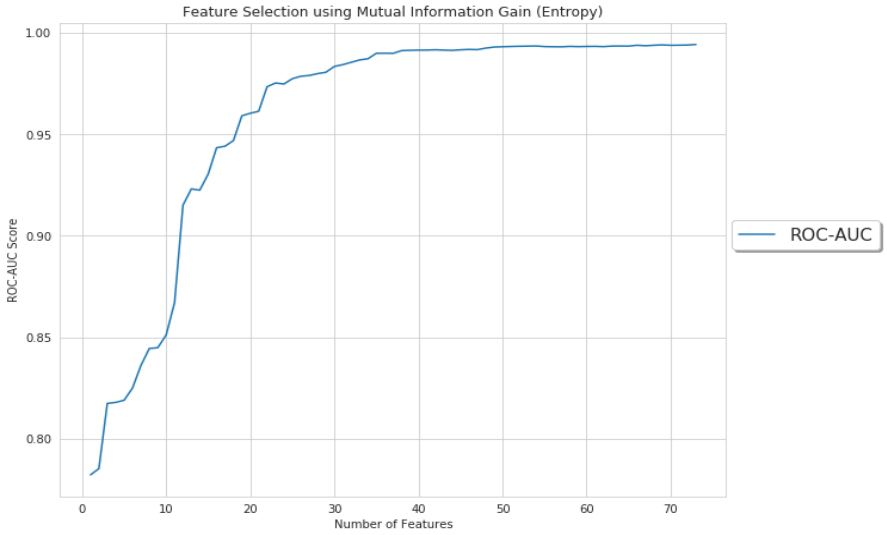

Fig. 3. ROC-AUC Score vs Number of Features using Mutual Information Gain

Feature Selection Based on Univariate ROC-AUC Score: In the incremental feature selection setup using Univariate ROC-AUC Score, the performance metrics also show a perceptive pattern (see Fig. 4 and Fig. 5). Distinguishably, the highest number

of features using Univariate ROC-AUC Score is 38, which is not the same as other setups (73 features). The reason is that the ROC-AUC scores of all features are not greater than 0.5. Only 38 features, considered in this setup, have an ROC-AUC score of greater than 0.5. So, those features are considered.

According to Fig. 4, as the number of features increases, the performances of the model also increases. In the range between 30-35 features, the performances of the model go on to almost constant as the lines proceed horizontally. This pattern holds true for ROC-AUC scores also (see Fig. 5).

This trend also signifies that all the 73 API Calls are not indicative of Android malware detection. Mostly, 30-35 features (API Calls) are responsible for impacting the performances of malware detection.

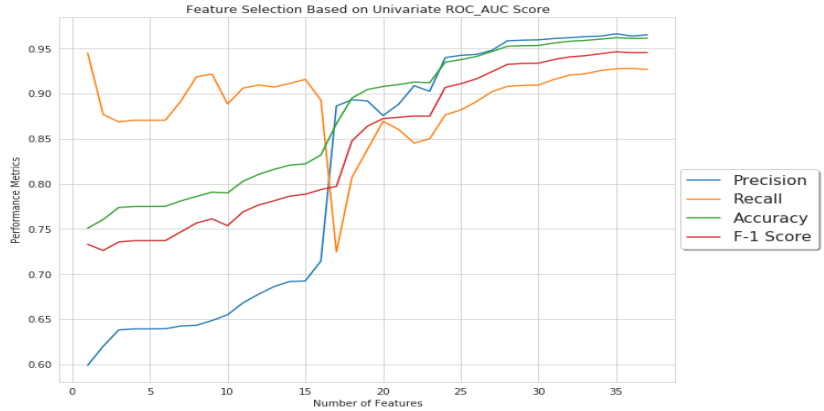

Fig. 4. Performance Metrics vs Number of Features using Univariate ROC-AUC Score

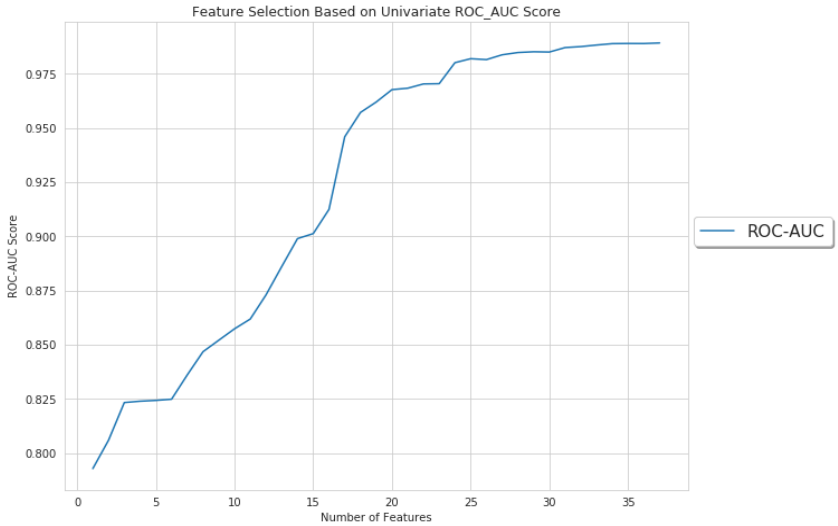

Fig. 5. ROC-AUC Score vs Number of Features using Univariate ROC-AUC Score

Feature Selection using SelectKBest (Chi-Squared): In the incremental feature selection setup using SelectKBest (Chi-Squared), the performance metrics also demonstrate an observant pattern (see Fig. 6 and Fig. 7).

According to Fig. 6 and Fig. 7, the optimal range of features lies in between 40-42 features as the line goes horizontally from that range.

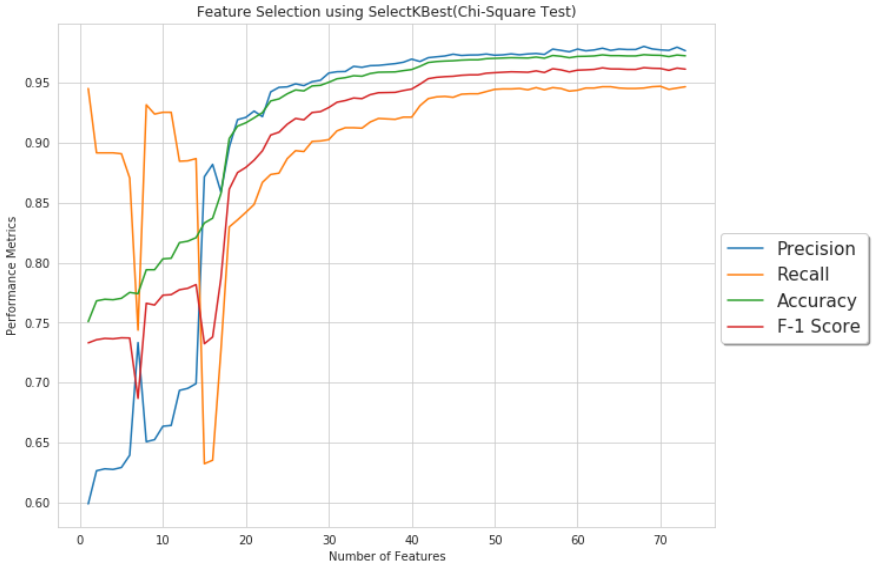

Fig. 6. Performance Metrics vs Number of Features using SelectKBest (Chi-Squared)

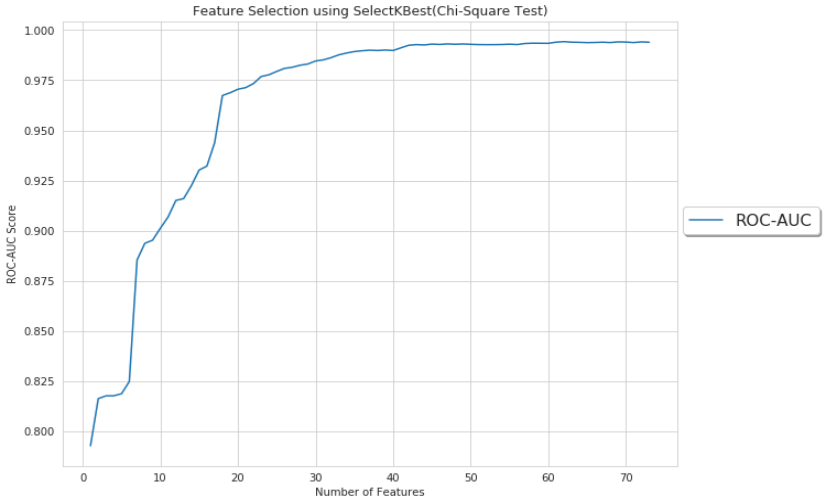

Fig. 7. ROC-AUC Score vs Number of Features using SelectKBest (Chi-Squared)

Feature Selection using Recursive Feature Elimination (RFE) with Gradient Boosting Classifier: In the incremental feature selection setup using the RFE with Gradient Boosting Classifier, the optimal range of features lies in between 24-30 features as the performance metrics are almost identical to the 73 features' performances. After the optimal range, the line goes almost horizontally (see Fig. 8 and Fig. 9).

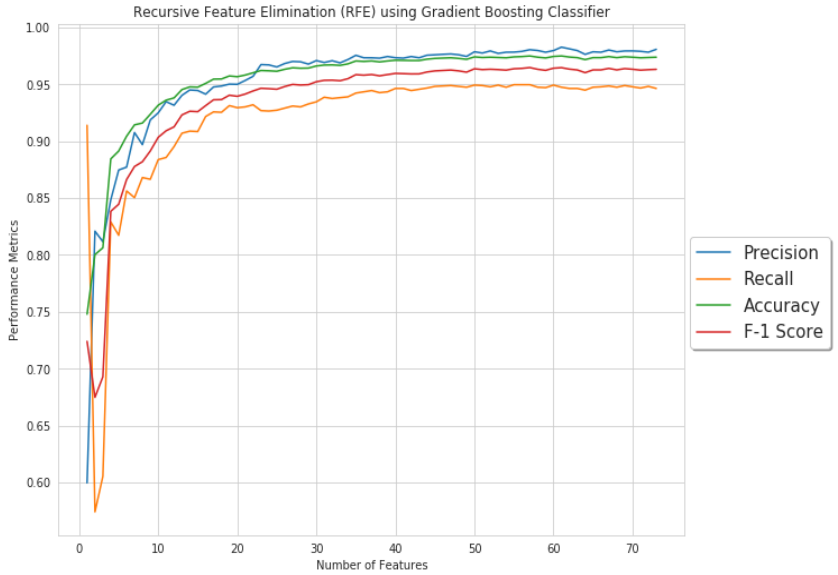

Fig. 8. Performance Metrics vs Number of Features using RFE (Gradient Boosting Classifier)

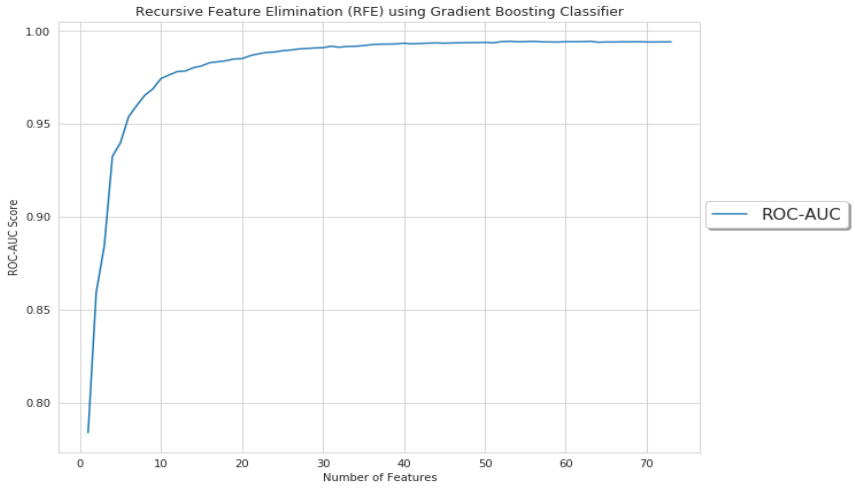

Fig. 9. ROC-AUC Score vs Number of Features using RFE (Gradient Boosting Classifier)

Feature Selection using Recursive Feature Elimination (RFE) with Random Forest Classifier: In the incremental feature selection setup using the RFE with Gradient Boosting Classifier, the optimal range of features lies in between 20-25 features as the performance metrics are almost identical to the 73 features' performances. After the optimal range, the line goes almost horizontally (see Fig. 10 and Fig. 11).

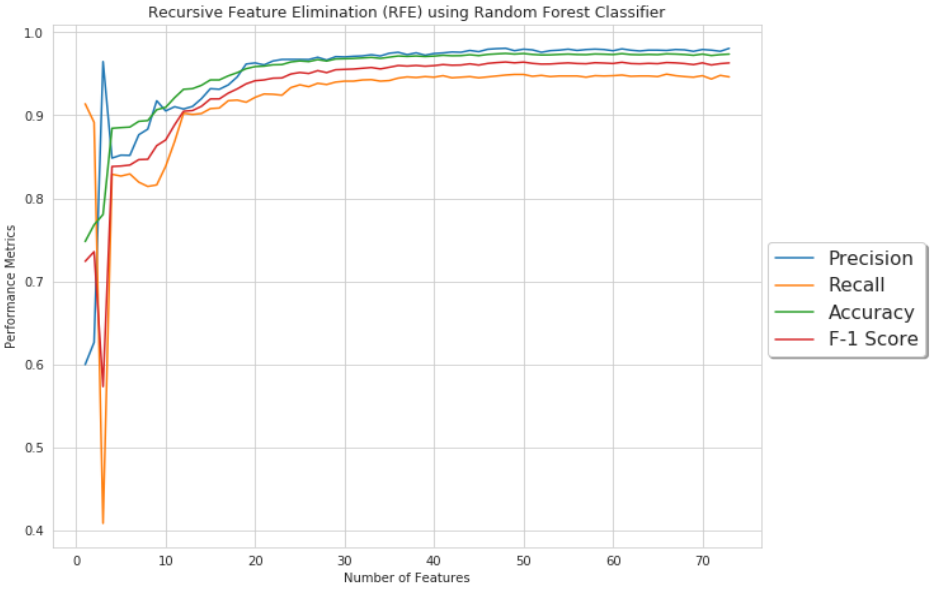

Fig. 10. Performance Metrics vs Number of Features using RFE (Random Forest Classifier)

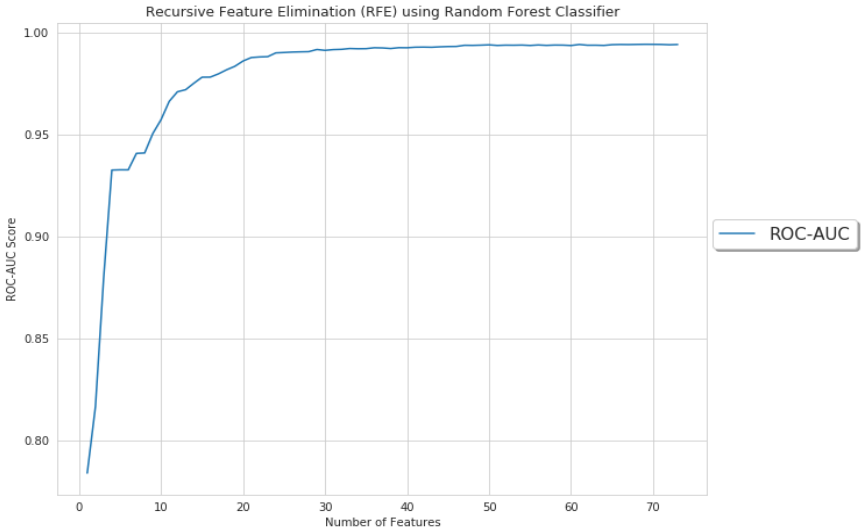

Fig. 11. ROC-AUC Score vs Number of Features using RFE (Random Forest Classifier)

3.3 Evaluation of Important API Calls (RQ2)

According to different feature selection techniques, RFE with Random Forest (RF) Classifier provides the lowest optimal range: 20-25 features. Therefore, the final evaluation of Important API Calls (IAC) is performed on those reduced feature sets.

Performance Evaluation of Important API Calls:

Table 1 shows the comparative performance among all the features and 20-30 feature sets. The result shows that using important 30 features, the accuracy, recall, f-1 score, and AUC are close enough and the precision is higher with respect to all the 73 features. Also, for the respective 28, 26, 24, 22, 20 features, the performance of Android malware detection remains close to the performance of 73 features. Therefore, the feature selection technique – RFE with Random Forest Classifier is suitable for identifying important API Calls in Android malware detection.

Table 1. Performance Evaluation of Important API Calls

Feature Selection Technique	# of API Calls	Accuracy (%)	Precision (%)	Recall (%)	F-1 Score	AUC
All Features	73	98.08	97.38	96.17	0.974	0.996
RFE with Random Forest Classifier	30	97.35	97.55	95.23	0.964	0.994
	28	97.24	96.64	94.50	0.956	0.993
	26	97.08	97.29	94.75	0.960	0.993
	24	96.75	96.64	94.50	0.956	0.993
	22	96.38	96.38	93.72	0.951	0.992
	20	96.30	96.67	93.22	0.950	0.991

Execution Time of Important API Calls:

Table 2 shows the comparative execution time of model training and testing. The result shows that using important 20-30 features, the execution time of model training and testing are substantially lower with respect to all the 73 features. For large data sets, this time would be substantially higher.

Table 2. Execution Time of Important API Calls

# of API Calls	Execution Time (ms)
73	6.48
30	4.38
28	4.31
26	4.26
24	4.12
22	4.04
20	3.95

Comparison with Existing Works:

Table 3 shows the comparative analysis of the detection rate using Important 20 API Calls – IAC (20) with respect to some existing works on Android malware detection. IAC (20) outperformed all the existing works except two.

Table 3. Comparison with the Existing Works

-	Detection Rate (%)
IAC (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%
Wang et al. [17]	94.62%

3.4 Critical API Calls in Android Malware Detection:

Table 3 shows the 30 critical API Calls in Malware Detection. These API Calls are derived from the feature selection technique – RFE with Random Forest Classifier.

Table 4. Critical API Calls in Android Malware Detection

Critical API Calls		
transact	android.content.pm.Signature	Ljava.lang.Class.getMethod
onServiceConnected	android.telephony.SmsManager	android.telephony.gsm.SmsManager
bindService	ClassLoader	attachInterface
TelephonyManager.getSubscriberId	Landroid.content.Context.registerReceiver	Ljava.lang.Object.getClass
ServiceConnection	Ljava.lang.Class.getField	abortBroadcast
android.os.Binder	Landroid.content.Context.unregisterReceiver	TelephonyManager.getDeviceId
Ljava.lang.Class.getCanonicalName	Ljava.lang.Class.getDeclaredField	HttpUriRequest
Ljava.lang.Class.getMethods	android.content.pm.PackageInfo	Ljava.net.URLDecoder
Ljava.lang.Class.cast	TelephonyManager.getLine1Number	TelephonyManager.getNetworkOperator
Runtime.exec	HttpGet.init	onBind

4 Limitation

In this study, only Drebin data sets 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.

5 Related Work

Several works dealt with API Calls in Android malware detection. Drebin incorporated API Calls along with other features and obtained an accuracy of 93.90% in malware detection [10]. Yerima et al. incorporated API Calls and Permissions with Bayesian Classifier and attained 92.1% accuracy [11]. In another work, they achieved an accuracy of 97.5% and an AUC of 0.953 by using a composite parallel classifier approach [12]. Using API Calls modeled with SVM (Support Vector Machine), Peiravian et al. gained an accuracy of 95.75% and an AUC of 0.957 [13]. DroidAPIMiner integrated API level features and reached an accuracy as high as 99% using the KNN classifier [14].

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 [15].

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 [16].

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 [17].

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

6 Conclusion

In this work, a comprehensive analysis is performed on API Calls in Android malware detection. First, various feature selection techniques are employed in an incremental fashion. According to the insights obtained from feature selection techniques,

an optimal number of features are selected. Afterward, evaluation of those features show that API Calls can be reduced without affecting performances so much. Therefore, reduced features set of API Calls would be useful in classifying Android malware considering performance and computational complexity.
In the future, the approach will be evaluated using different data sets. Also, other feature selection techniques using deep learning, feature importance, etc. will be employed.

References

1. Kraskov, A., Stögbauer, H., & Grassberger, P. (2011). Erratum: Estimating mutual information [Phys. Rev. E 69, 066138 (2004)]. Physical Review E, 83(1), 019903.
2. Ross, B. C. (2014). Mutual information between discrete and continuous data sets. PloS one, 9(2).
3. Cover, T. M., & Thomas, J. A. (2012). Elements of information theory. John Wiley & Sons.
4. Fawcett, T. (2006). An introduction to ROC analysis. Pattern recognition letters, 27(8), 861-874.
5. sklearn.feature_selection.RFE¶. (n.d.). Retrieved February 21, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html
6. Fraj, M. B. (2017, December 24). In Depth: Parameter tuning for Gradient Boosting. Retrieved February 1, 2020, from https://medium.com/all-things-ai/in-depth-parameter-tuning-for-gradient-boosting-3363992e9bae
7. Ho, T. K. (1995, August). Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278-282). IEEE.
8. sklearn.feature_selection.SelectKBest¶. (n.d.). Retrieved February 11, 2020, from http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
9. Pearson, K. (1900). 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, 50(302), 157-175.
10. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., & Siemens, C. E. R. T. (2014, February). Drebin: Effective and explainable detection of android malware in your pocket. In Ndss (Vol. 14, pp. 23-26).
11. 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.
12. 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.
13. 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.
14. 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.

15. 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.
16. 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.
17. 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.