DOI: https://doi.org/10.52756/ijerr.2023.v34spl.009 *Int. J. Exp. Res. Rev.*, Vol. 34: 86-96 (2023)

Original Article | Peer Reviewed | Open Access

International Journal of Experimental Research and Review (IJERR)
© Copyright by International Academic Publishing House (IAPH)
ISSN: 2455-4855 (Online)
www.iaph.in

An efficient android malware detection method using BorutaShap algorithm

Sandeep Sharma[1*], Prachi[1], Rita Chhikara[1] and Kavita Khanna[2]

[1]Department of Computer Science, The North Cap University, India;
[2]Delhi Skill and Entrepreneurship University, India

E-mail/Orcid Id:

SS, sandeep19csd004@ncuindia.edu; *P,* prachi@ncuindia.edu, https://orcid.org/0000-0002-6241-7659; *RC,* ritachhikara@ncuindia.edu, https://orcid.org/0000-0001-9537-3907; *KK,* kavita.khanna@dseu.ac.in, https://orcid.org/0000-0001-5745-1206

Article History:

Received: 20th Jul., 2023
Accepted: 08th Oct., 2023
Published: 30th Oct., 2023

Keywords:

Android, Feature Selection, Feature Extraction, Malware Detection, Machine Learning, Optimized features

How to cite this Article:
Sandeep Sharma, Prachi, Rita Chhikara and Kavita Khanna (2023). An efficient Android Malware Detection Method using Borutashap Algorithm. *International Journal of Experimental Research and Review*, *34*(Spl.), 86-96.
DOI : https://doi.org/10.52756/

Abstract: The Android operating system captures the largest global smartphone market share. However, its popularity and open-source nature have garnered the attention of cybercriminals. The landscape of Android malware has evolved significantly over time. Traditional techniques for detecting Android malware are encountering difficulties in keeping up with this evolution. Specifically, methods that rely on extracting various features from Android applications are becoming difficult to implement as high-dimensional feature sets incur huge computational overheads when employed with machine learning algorithms. Therefore, this research proposes using Bortua and BorutaShap feature selection algorithms to choose features that contribute to detecting malicious Android applications. It uses static and dynamic features of Android applications to create a detection model for verification and evaluation of the mentioned algorithms. Experimental results showed that Bortua and BorutaShap algorithms offer promising results by achieving the highest accuracy of approximately 99%.

Introduction

Android operating system (OS) is the extensively used OS in smart devices such as smartphones, tablets, smart TV's, Android Auto and Smartwatches. First unveiled in 2007, it quickly became popular because it is open-source, commercially backed, and developed by Google. According to Statista (2003), Android is the main mobile OS with a 71.8% market share in the fourth quarter of 2022. This huge user base makes it a very lucrative target for attackers. In addition to the Google Play Store, users can also get applications from other third-party marketplaces. The open-source nature of Android OS, large user base and availability of applications on various third-party stores make Android users more prone to various malware attacks. Android malware can steal information, cause disruption in services, gain unauthorized access, modify data, spam users, or, in general, interfere with the security of a user's device. According to Cyber Security News and Insights for Executives (2022), Android malware attacks are over 50 times higher than iOS malware attacks. Google describes Android malware as Potentially Harmful applications (PHAs). In 2018 research report by Google (Android, 2019), the PHAs are classified as Trojan, SMS fraud, phishing, hostile downloaders, spyware, privilege escalation, toll fraud, click fraud, backdoors and commercial spyware. Researchers have investigated several ways of detecting Android malware. These techniques are categorized as static and dynamic analysis-based techniques.

Static analysis-based detection methods reverse engineer the applications to retrieve static features such as permissions, intents, API calls, opcodes, etc (Zhang et al., 2020; Kazmi et al., 2023; Goel et al. 2023). Dynamic analysis-based detection methods execute applications on Android devices or emulators to observe the application's behavior. Mainly, the dynamic analysis methods (Khan and Jain, 2020; Martinelli et al., 2017; Vinod et al., 2019;

Xu et al., 2019; Li et al., 2019; Kaleem et al., 2023; Vivekanandan and Gunasekaran, 2023) use a simulated environment that triggers the activity of malicious code (Alzaylaee et al., 2020) and monitors the behavior of the application. These techniques are much more effective in the case of obfuscated malware or dynamically loaded code. Thus, it can also detect novel malware based on their real-time activities. However, they fail to offer complete code coverage.

An efficient and robust Android malware solution should consider static and dynamic features because they are key attributes defining an application as benign or malicious. However, the extracted features are frequently afflicted by high dimensionality, leading to a considerable increase in computational overhead. Also, many features are redundant and irrelevant in nature. They increase the complexity and decrease the performance of a classification model. Therefore, this work proposes an Android malware detection model based on a few important static as well as dynamic features that can effectively differentiate malicious applications from benign ones.

Contributions to this work are -

(1) Extract hybrid features from benign and malicious APKs.

(2) Reduce the dimensionality using Boruta, BorutaShap and Gini importance methods.

(3) Detect the malware and calculate Android malware detection accuracy using Random Forest algorithm and AdaBoost algorithm.

The rest of the paper is organized as follows:

Section 2 discusses the related work and section 3 presents the proposed methodology, section 4 includes experimental results and discussion and section 5 concludes the paper.

Related Work

Several researchers have explored the application of feature selection in the context of Android malware detection. Here is a summary of their findings and methods:

1. Deepa et al. (2015): They used the correlation feature selection method and achieved an accuracy of 88.75% in Android malware detection by mining the top few features.

2. Zhao et al. (2015): They introduced a novel feature selection technique that considered the frequency of features in benign and malicious Android samples. Their method achieved up to 98% accuracy by selecting distinct features.

3. Wen and Yu (2017): They extracted hybrid features from applications and used Principal Component Analysis and Relief to extract promising features. Their method achieved 95.2% accuracy on malicious applications.

4. Altaher and Barukab (2017): They extracted permissions and API calls and applied filter-based feature selection techniques to rank features based on importance in distinguishing malicious samples from benign ones.

5. Bhattacharya and Goswami (2018): They presented a feature selection method and obtained 87.8% and 97.9% accuracy rates on two different datasets, claiming that their technique can perform well on large datasets.

6. Peynirci et al. (2020): They selected features with the highest Inverse Document Frequency (IDF) in benign applications and the lowest IDF in malicious applications, achieving above 99% accuracy on three different datasets.

7. Dhalaria and Gandotra (2020): They extracted hybrid features from Android applications and used the Chi-Square feature selection method to select important features for detecting malicious Android applications.

8. Sahin et al. (2021): They applied eight different feature selection techniques and demonstrated that feature selection methods from text classification studies can improve classification model performance.

9. Kouliaridis et al. (2021): They used PCA and t-SNE with multiple machine learning algorithms to select features from the entire feature set and achieved an accuracy of 91.7%.

These studies highlight the importance of feature selection in enhancing the accuracy and efficiency of Android malware detection algorithms.

Proposed Method

The method followed in this work to detect malware and measure the accuracy of different algorithms can be divided into 3 steps. First, to create a dataset of static and dynamic features of APKs using tools such as MobSF, Strace and Androguard. Then, a feature set of corresponding features from APKs is created in which if a feature/attribute is present in a particular APK, then it is marked as 1 otherwise 0. Second, Boruta, BorutaShap and Gini Importance feature selection algorithms are applied to extract the important features from the feature set. This improves the performance of the classification model by removing redundant information. Lastly, Random Forest and AdaBoost classifiers are applied to the selected features to find the best possible approach for malware detection.

Dataset Collection

This step collects benign and malicious Android samples from two major sources: malware samples from AndroZoo repository (Allix et al., 2016) and benign samples from Google Play Store. This work uses AndroZoo to collect malware samples because it is a comprehensive repository containing samples from various sources. Later on, samples are executed within Genymotion emulator and their dynamic features, such as system calls, are retrieved through Mobile Security Framework (MobSF) (Abraham, 2022) and the strace tool. Further, static features are extracted with the help of MobSF and Androguard (Androguard, 2019). The Androguard tool reverse-engineers the samples to obtain manifest.xml and classes.dex files. The following static features are obtained from these samples: opcode, permission, API call and intent filter.

Feature Selection - Boruta and BorutaShap Algorithm

Feature selection techniques are one of the best ways to handle the curse of dimensionality and reduce over-fitting in a machine-learning model. It makes the model simple and increases the interpretation of the model, highlighting the most informative features. This research work uses Boruta algorithm to elect the most contributory features.

Boruta Algorithm works on the concept of creating duplicates referred to as shadow features of all independent features, as shown in Figure 2. The values are shuffled after that to remove the correlation between independent and dependent features. The algorithm works on the concept of top-down search. It compares the importance of original attributes with shadow features created through permutation and gradually eliminates the features that are not important. The features that are better than shadow features are retained.

The algorithm then computes a Z score, finds the greatest Z score, and tags variables as 'unimportant' or 'important' depending on whether they are significantly lower or higher than the highest Z score. This process is repeated until all attributes are either tagged 'unimportant' or 'important'.

Boruta can be combined with SHAP (SHapley Additive exPlanations) in a machine learning pipeline for feature selection and interpretation of the model. SHAP values are used to explain the contribution of each feature in a predictive model. They are used to give a summary of the importance of features.

Performance Metrics

This work uses the following metrics for evaluating the performance of ML algorithms.

Precision

This metric shows that out of predicted positive values (predicted malware), how many values were actually positive (actual malware).

$$\text{Precision} = \frac{True\ Positives}{\text{True Positives} + \text{False Positives}}$$

Recall

Recall means ratio of predicted positives (predicted malware) out of total actual positives.

$$\text{Recall} = \frac{True\ Positives}{\text{True Positives} + \text{False Negatives}}$$

F1-score

F1-score is used to get a balanced metric out of precision and recall.

$$\text{F1} - \text{Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Accuracy

Accuracy defines the overall correctness of our model. It is the ratio of correct predictions out of all predictions made by the model.

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FP} + \text{TN} + \text{FN})}$$

Experimental Results and Analysis

This section includes various experiments and tests conducted on the dataset mentioned in the previous section, which includes 102 dynamic features and 915 static features on 190 APK samples. To maintain uniformity, the dataset is split into train and test for all the experiments at 60%:40%. The feature selection techniques Boruta, BorutaShap and Gini Importance filter the dynamic and static features that can detect Android Malware efficiently. The performance of selected features is compared with Random Forest and AdaBoost. The results and analysis of various experiments are as follows-

Results of Feature Selection

As reported in Table 1, the original dimensionality is significantly reduced with all the feature selection techniques with good accuracy. However, the BorutaShap with Random Forest Classifier outperforms all the other techniques. The number of features is significantly reduced from 13102 to 30 for dynamic features and 915 to 74 for static features by Boruta. BorutaShap obtains the best accuracy from Table 1 as a Feature Selector with a Random Forest Classifier. The BorutaShap has the advantage over other methods applied in this study as it projects a strong addictive explanation of features, which is the strength of the SHAP method and at the same time, it has the robustness of the Boruta algorithm. This ensures that only significant variables remain on the set.

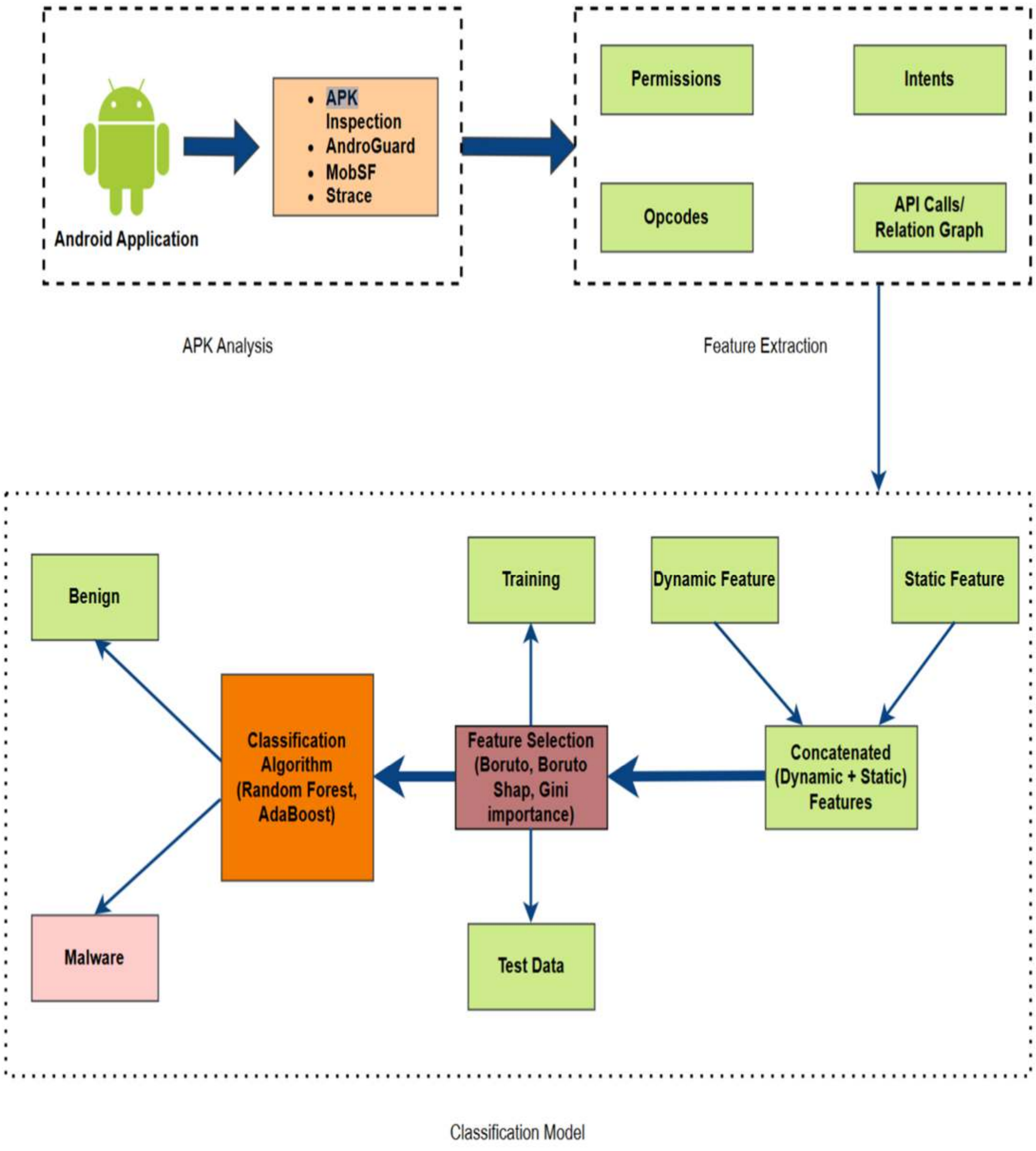
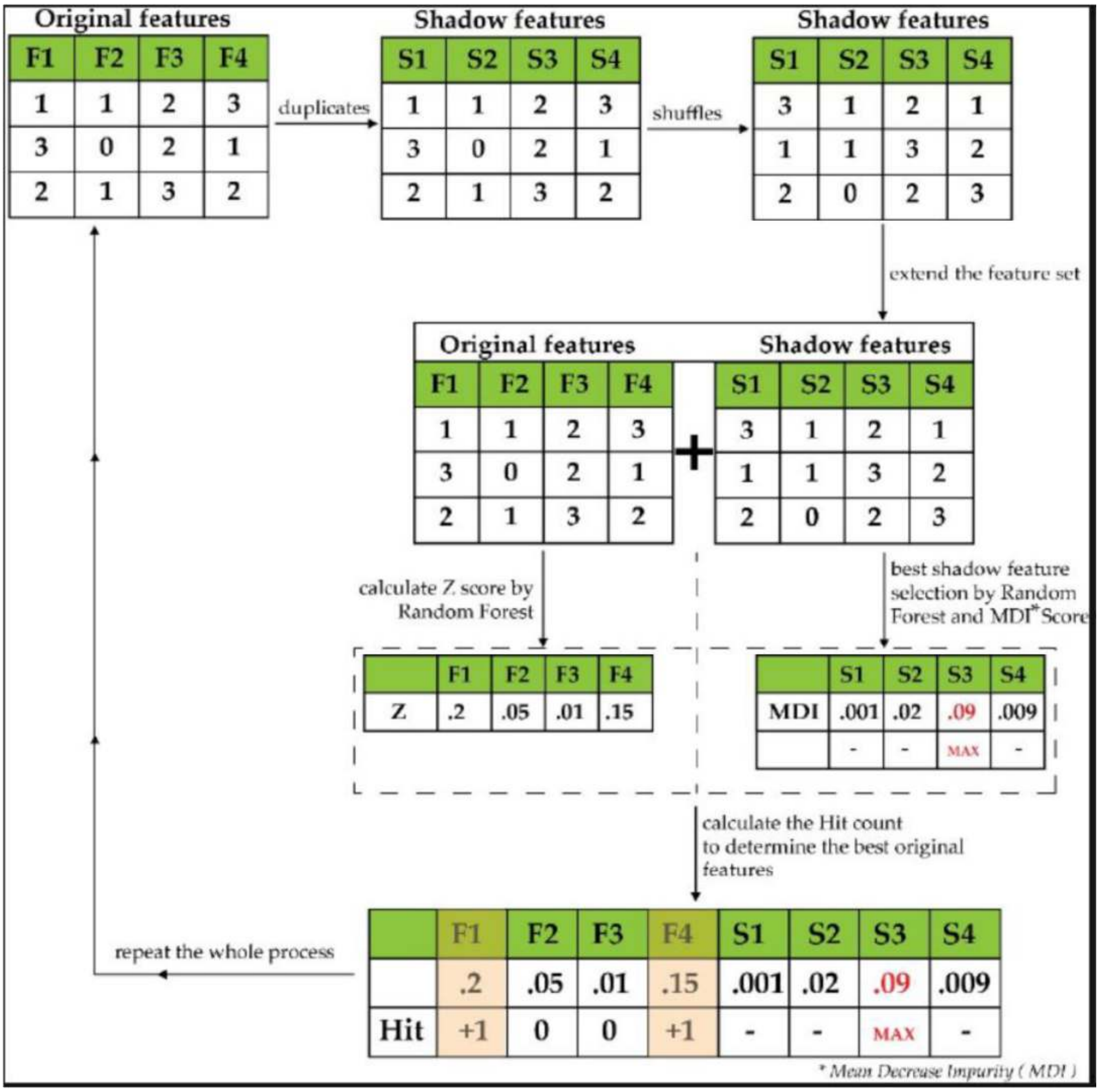


Figure 1. Proposed model for feature extraction and analysis


Original features
duplicates
Shadow features
shuffles
Shadow features
extend the feature set
Original features
Shadow features
+
calculate Z score by Random Forest
best shadow feature selection by Random Forest and MDI* Score
calculate the Hit count to determine the best original features
repeat the whole process
* Mean Decrease Impurity (MDI)

F1	F2	F3	F4
1	1	2	3
3	0	2	1
2	1	3	2

S1	S2	S3	S4
1	1	2	3
3	0	2	1
2	1	3	2

S1	S2	S3	S4
3	1	2	1
1	1	3	2
2	0	2	3

F1	F2	F3	F4	S1	S2	S3	S4
1	1	2	3	3	1	2	1
3	0	2	1	1	1	3	2
2	1	3	2	2	0	2	3

	F1	F2	F3	F4
Z	.2	.05	.01	.15

	S1	S2	S3	S4
MDI	.001	.02	.09	.009
	-	-	MAX	-

	F1	F2	F3	F4	S1	S2	S3	S4
	.2	.05	.01	.15	.001	.02	.09	.009
Hit	+1	0	0	+1	-	-	MAX	-

Figure 2. Boruta algorithm Steps

Table 1. Comparison of the number of static and dynamic features selected

Classifier	Feature Selecctor	No. of Dynamic Features Selected	Accuracy (%)	Number of Static Features Selected	Accuracy (%)
Random Forest	Boruta	30	88	74	98
	BorutaShap	30	90	74	98.6
	Gini Importance	32	84	80	98
AdaBoost	Boruta	30	70	74	85
	BorutaShap	30	72.3	74	76.4
	Gini Importance	32	60.5	80	69.3

Table 2. Comparison of different Feature Selection models

Classifier	Type of Feature Selector	Accuracy (%)	Precision	Recall	F1-Score
Random Forest	Without Feature Selection	95.4	0-94.0% 1-94.4%	0-95.0% 1-94.6%	0-95.2% 1-93.7%
	Boruta	98.6	0-99.3% 1-97.0%	0-98.0% 1-99.4%	0-99.0% 1-98.0%
	BorutaShap	98.9	0-99.4% 1-98.0%	0-98.0% 1-99.2%	0-99.0% 1-99.0%
	Gini Importance	98.6	0-99.0% 1-94.0%	0-96.0% 1-99.9%	0-98.0% 1-97.0%
AdaBoost	Without Feature Selection	82.5	0-80.0% 1-78.4%	0-79.0% 1-76.8%	0-81.1% 1-77.3%
	Boruta	85.9	0-92.0% 1-75.0%	0-87.0% 1-83.0%	0-89.0% 1-79.0%
	BorutaShap	78.9	0-86% 1-68%	0-81% 1-71%	0-83% 1-71%
	Gini Importance	70.1	0-72% 1-65%	0-83% 1-50%	0-77% 1-56%

Comparison of Random Forest performance with different feature selection techniques

The three feature selection techniques Boruta, BorutaShap and Gini Importance, are evaluated by two popular classification algorithms Random Forest and AdaBoost. As observed from Table 2 Random Forest with BorutaShap feature selector outperforms all other combinations of feature selector and classification algorithm. Random Forest works on the concept of an ensemble of decision trees through bagging. This reduces the variance of the single decision tree and overcomes overfitting in the model. The highest accuracy as reported in Table 2, is 98.9% and Precision, Recall and F1-score of approx. 99%. On the other hand, AdaBoost algorithm with the Gini importance feature selector does not give good results. The lowest accuracy is 70%, reported by Adaboost+Gini Importance.

The Figure 3 displays the feature importance of all features through Boruta with Shap as feature selector. Figure 4 and Figure 5 project the boxplot of top 10 dynamic features and static features respectively. Higher value of z-score represents the significant importance of the feature.

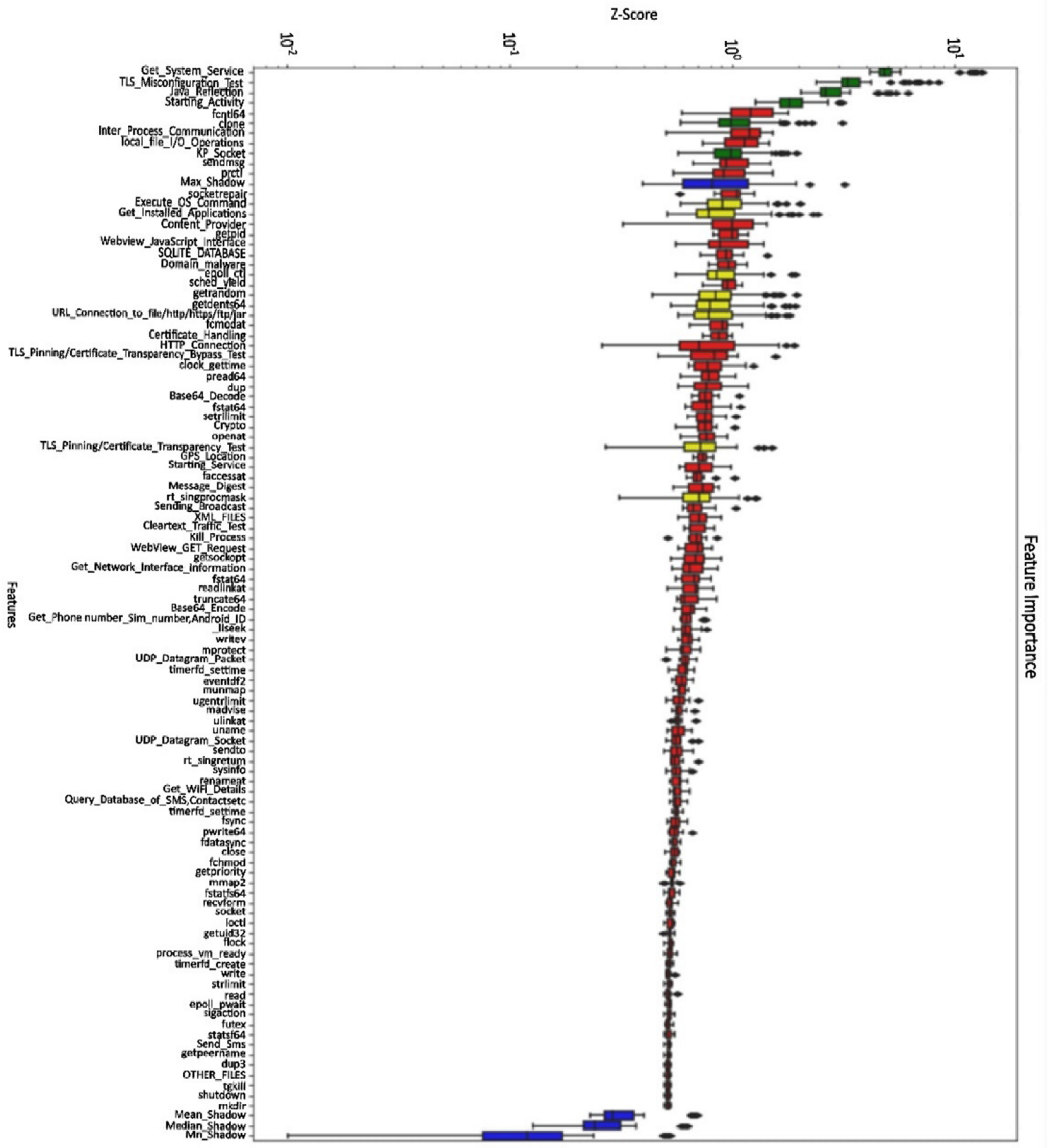


Figure 3. Feature importance graph for all the features

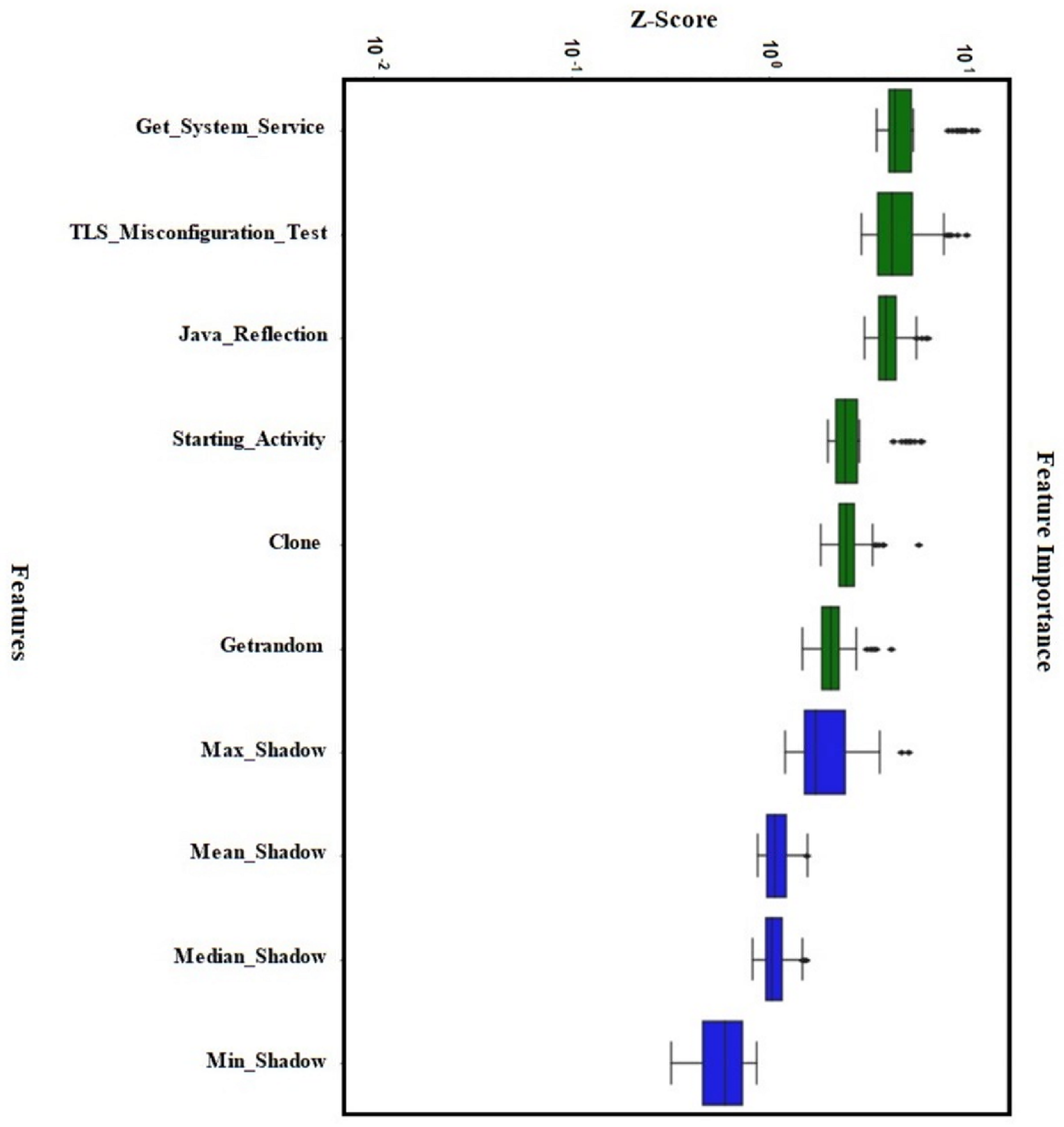


Figure 4. Feature importance graph of selected dynamic features using BorutaShap

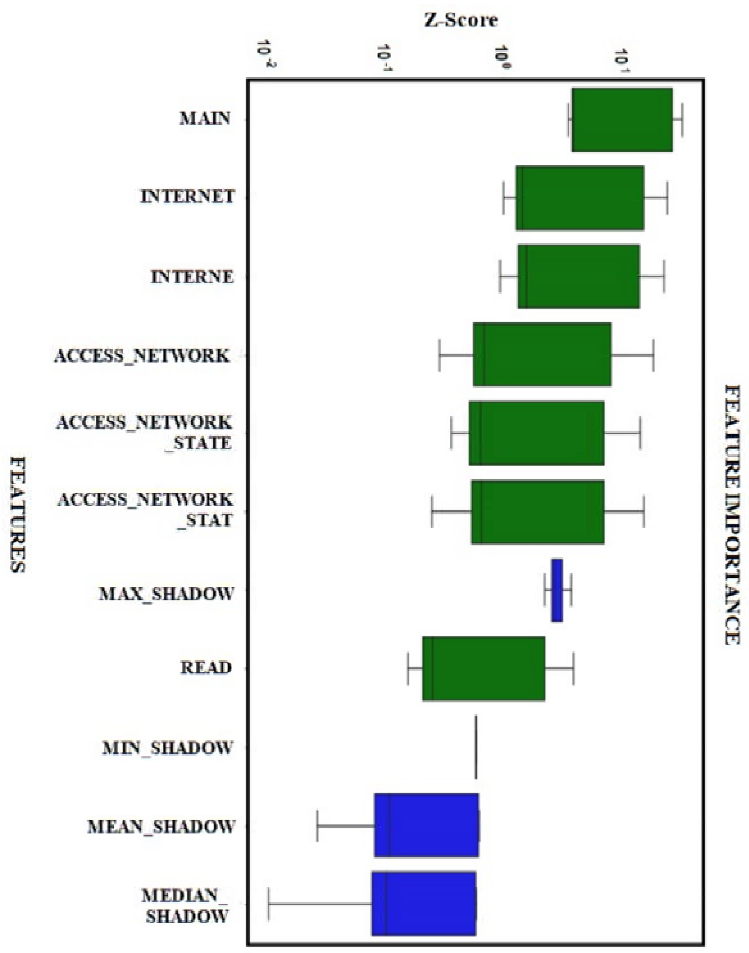


Figure 5. Feature importance graph of selected static features using BorutaShap

Conclusion

This work proposes Boruta and Boruta Shap feature selection algorithm for detecting malicious Android applications. It reduces the count of features and verifies the performance of Boruta and Boruta Shap algorithms in Android malware. The experimental results have been performed under different scenarios, static and dynamic features were evaluated individually as well as in an integrated manner. Results proved that the Boruta Shap algorithm is more effective than other state-of-art algorithms in decreasing feature dimension and increasing accuracy. The performance is determined through various metrics and classification algorithms: Random Forest and AdaBoost.

Further work will implement this framework on a benchmark dataset to compare results with state-of-art works in the literature. Further, in addition to binary classification, multi-class classification will be performed on malicious samples to map them into their respective families.

References

Abraham, A. (2022). Mobile Security Framework (MobSF). *https://github.com/MobSF/Mobile-Security-Framework-MobSF*

Allix, K., Bissyandé, T. F., Klein, J., & Le Traon, Y. (2016). Androzoo: Collecting millions of android apps for the research community. *In Proceedings of the 13th International Conference on Mining Software Repositories,* pp. 468-471. https://doi.org/10.1145/2901739.2903508

Altaher, A., & Barukab, O.M. (2017). Intelligent hybrid approach for Android malware detection based on permissions and API calls. *International Journal of Advanced Computer Science and Applications*, *8*(6), 60-67. https://doi.org/10.14569/IJACSA.2017.080608

Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2020). DL-Droid: Deep learning based android malware detection using real devices. *Computers & Security*, *89*, 101663.

https://doi.org/10.1016/j.cose.2019.101663

AndroGuard. (2019). Reverse engineering and pentesting for Android applications. *https://pypi.org/project/androguard/*

Android. (2019). Android Security & Privacy 2018 year In Review. *https://source.android.com/security/reports/Google_Android_Security _2018_Report_Final.pdf.*

Bhattacharya, A., & Goswami, R. T. (2018). A hybrid community based rough set feature selection technique in android malware detection. Springer Singapore, *In Smart Trends in Systems, Security and Sustainability: Proceedings of WS4, 2017*, 249-258. https://doi.org/10.1007/978-981-10-6916-1_23

Cyber Security News and Insights for Executives. (2022). 10 eye-opening mobile malware statistics to know. *https://www.cybertalk.org/2022/06/10/10-eye-opening-mobile-malware-statistics-to-know/*

Deepa, K., Radhamani, G., & Vinod, P. (2015). Investigation of feature selection methods for android malware analysis. *Procedia Computer Science, 46*, 841-848. https://doi.org/10.1016/j.procs.2015.02.153

Dhalaria, M., & Gandotra, E. (2020). Android malware detection using chi-square feature selection and ensemble learning method. *IEEE, In 2020 Sixth International conference on parallel, distributed and grid computing* (PDGC), pp. 36-41. https://doi.org/10.1109/PDGC50313.2020.9315818

Goel, A., Wasim, J., & Srivastava, P. (2023). A Noise reduction in the medical images using hybrid combination of filters with nature-inspired Black Widow Optimization Algorithm. *International Journal of Experimental Research and Review*, *30*, 433-441. https://doi.org/10.52756/ijerr.2023.v30.040

Kaleem, W., Sajid, M., & Rajak, R. (2023). Salp Swarm Algorithm to solve Cryptographic Key Generation problem for Cloud computing. *International Journal of Experimental Research and Review*, *31*(Spl Volume), 85-97. https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009

Kazmi, S., Singh, M., & Pal, S. (2023). Image Retrieval Performance Tuning Using Optimization Algorithms. *International Journal of Experimental Research and Review*, *33*, 8-17. https://doi.org/10.52756/ijerr.2023.v33spl.002

Khan, M.A., & Jain, M.K. (2020). Feature Point Detection for Repacked Android Apps. *Intelligent Automation & Soft Computing*, *26*(6), 1359–1373. https://doi.org/10.32604/iasc.2020.013849

Kouliaridis, V., Potha, N., & Kambourakis, G. (2021). Improving android malware detection through dimensionality reduction techniques. Springer International Publishing, *In Machine Learning for Networking: Third International Conference, MLN* 2020, Paris, France, November 24–26, 2020, Revised Selected Papers 3, pp. 57-72. https://doi.org/10.1007/978-3-030-70866-5_4

Li, Y., Xu, G., Xian, H., Rao, L., & Shi, J. (2019). Novel Android Malware Detection Method Based on Multi-dimensional Hybrid Features Extraction and Analysis. *Intelligent Automation & Soft Computing*, *25*(3), 637-647. https://doi.org/10.31209/2019.100000118

Martinelli, F., Marulli, F., & Mercaldo, F. (2017). Evaluating convolutional neural network for effective mobile malware detection. *Procedia Computer Science*, *112*, 2372-2381. https://doi.org/10.1016/j.procs.2017.08.216

Peynirci, G., Eminağaoğlu, M., & Karabulut, K. (2020). Feature selection for malware detection on the android platform based on differences of IDF values. *Journal of Computer Science and Technology*, *35*, 946-962. https://doi.org/10.1007/s11390-020-9323-x

Şahin, D. Ö., Kural, O. E., Akleylek, S., & Kılıç, E. (2021). A novel Android malware detection system: adaption of filter-based feature selection methods. *Journal of Ambient Intelligence and Humanized Computing*, *14*, 1243-1257. https://doi.org/10.1007/s12652-021-03376-6

Statista (2023). Mobile operating systems' market share worldwide from 1st quarter 2009 to 4th quarter 2022.

Vinod, P., Zemmari, A., & Conti, M. (2019). A machine learning based approach to detect malicious android apps using discriminant system calls. *Future Generation Computer Systems*, *94*, 333-350. https://doi.org/10.1016/j.future.2018.11.021

Vivekanandan, S., & Gunasekaran, G. (2023). A Computation of Frequent Itemset using Matrix Based Apriori Algorithm. *International Journal of Experimental Research and Review*, *30*, 247-256. https://doi.org/10.52756/ijerr.2023.v30.022

Wen, L., & Yu, H. (2017). An Android malware detection system based on machine learning. AIP Publishing, *In AIP Conference Proceedings,* Vol. 1864, No. 1. https://doi.org/10.1063/1.4992953

Xu, K., Li, Y., Deng, R., Chen, K., & Xu, J. (2019). Droidevolver: Self-evolving android malware detection system. IEEE, In *2019 IEEE European Symposium on Security and Privacy* (Euro S & P). pp. 47-62. https://doi.org/10.1109/EuroSP.2019.00014

Zhang, X., Zhang, Y., Zhong, M., Ding, D., Cao, Y., Zhang, Y., Zhang, M., & Yang, M. (2020). Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware. *In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security,* pp. 757-770. https://doi.org/10.1145/3372297.3417291

Zhao, K., Zhang, D., Su, X., & Li, W. (2015). Fest: A feature extraction and selection tool for Android malware detection. IEEE, *In 2015 IEEE symposium on computers and communication* (ISCC), pp. 714-720. https://doi.org/10.1109/ISCC.2015.7405598

How to cite this Article:

Sandeep Sharma, Prachi, Rita Chhikara and Kavita Khanna (2023). An efficient Android Malware Detection Method using Borutashap Algorithm. *International Journal of Experimental Research and Review*, *34*(Spl.), 86-96.

DOI: https://doi.org/10.52756/ijerr.2023.v34spl.009