International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 5, Issue 7, pp 56-61, July-2021

https://doi.org/10.47001/IRJIET/2021.507010

Opcode-Based Android Malware Detection Using Machine Learning Techniques

[1]Sonal Pandey, [2]Ram Lal

[1]NITTTR Chandigarh, India. E-mail: pandey.sonal88@gmail.com
[2]Computer Service Centre, IIT Delhi, India. E-mail: ramlal@cc.iitd.ac.in

***Abstract* - Today Android applications are widely used by billions of users to perform different activity. So malware target the Android phone frequently. The new malware sample is a major issue and signature based technique are unable to find out new malware sample. In this paper author will appliance an approach to detect the unfamiliar Android malware using machine learning techniques with a high detection rate. We adopt sampling technique based on the sensitive opcodes sequence. Finally, we evaluate our method on AndroZoo dataset (15000 malware and 15000 benign Apks), and select the top 20 malware families for experiments. The experimental results show that the Total Accuracy 95.3%, 92.16%, and 92.6% with random forest, XGBosst, and Decision tree.**

Keywords: Malware, Metamorphic malware, Android, Machine Learning.

I. INTRODUCTION

Any application or program or code which is planned deliberately to get through the entrance framework or the approval interaction and acquires a state where it can make unacceptable moves is known as malevolent programming or malware. Malware is harmful software that infiltrates your system and performs undesirable tasks while compromising the computer system's security policy in expressions of data confidentiality, integrity, and availability. It has the ability to change or remove any software package from the system in direction to intentionally disrupt the system's essential functionality [13]. Trojan Horse, spyware, adware, and worms are all examples of malware.

II. MOTIVATION

Now a day's number of active user are increase day by day on mobile phone. Most of the smart phone is depend upon the android platform. Android is an influential operating system and it supports a large amount of applications on Smartphones. These applications are more comfortable and innovative for users. It is engaged approximately 75% of shares in the worldwide market by the end of 2020. As of late, Android has overwhelmed other mobile operating systems to become perhaps the most generally utilized platforms[1], which is why malware authors are increasingly targeting the Android platform.

III. OBJECTIVE

- To analyze the existing malware detection techniques.
- To extract essential features from existing dataset by performing static analysis.
- To use supervised machine learning model for classification of malware.
- To evaluate and compare the performance of proposed approach with existing approaches.

IV. LITERATURE REVIEW

A writing survey is critical to any examination project since it recognizes the requirement for the work. Ongoing examinations on the utilization of machine learning calculations with static and dynamic investigation for distinguishing malware include:

Bilar [26] used an opcode frequency distribution to identify and distinguish between malware and benign. Bilar took a small dataset of 67 malware and 20 benign files for statistical analysis. Bilar examined 14 opcodes that are frequently observed in malware files. But he was unable to classify the second generation malware.

Rudy [33] foster a static malware location framework by utilizing text arrangement methods dependent on the show substance of the dubious applications [12]. Author use different text classification techniques (Naïve Bayes, Multinomial Naïve Bayes, C4.5, SVM, etc.) to apply different dataset. Author achieve malware detection accuracy rate between 94.0% and 99.3% and got highest accuracy through SVM (99.3%).

Cai et al. [42] proposed a novel dynamic application grouping strategy, to supplement existing methodologies. DroidCat accomplishes superior robustness than static methodologies just as dynamic methodologies depending on framework calls [11]. Author used a Random forest classifier

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 5, Issue 7, pp 56-61, July-2021

https://doi.org/10.47001/IRJIET/2021.507010

to classify the malevolent and benign application and got 94.5% accuracy.

Wang et al. [28] extracted used permissions and API calls from applications to train a model using AdaboostM1 classier. By evaluating their model on 1170 malware samples and 1205 benign samples, they claim a True Positive Rate of 99.6%.

Feizollah et al. [14] seemed the feasibility of unequivocal Intent and implied Intent for Android malware disclosure. The evaluation has been done on 5560 malware tests and 1846 liberal tests. Authors come to 91% precision by consuming-through Android Intent, 83% precision utilizing authorization and by joining made both the features they achieved the discovery rate of 95.5% [1].

Sun et al. [15] presented SigPID which is centered on the permission examination to identify noxious applications. Creators have extricated 135 permissions from the dataset but utilized as it were 34 permission (25% of add up to consents) to recognize between malicious and benign applications. Author utilized the SVM for model training and claims a precision of 93.62% for malware inside the dataset and 91.4% for obscure malware.

Xu et al. [38] proposed a strategy to identify noxious application known as ICC Detector which depends on the between segment correspondence related highlights. ICC Detector can screen the communication between the parts or cross-application limits [3].Authors utilized 5264 malware and 12026 favorable applications and got 97.4% discovery precision.

Wain et al. [18] In this paper, authors analyzed permissions present in the android application and developed a tool PScout that extracts permission specifications using static analysis.

Tao et al. [19] In this paper, authors taken into consideration hidden styles of malware in real-international Android applications. The authors extricated sensitive APIs which are applied in malware moreover actualized a mechanized malware discovery framework to become aware of difficult to understand android malware [1]. The authors carried out a complete studying approximately using 31185 type and 15336 malware assessments and were given 98.24 F1 scores.

Schultz et al. [25] introduced machine learning concept for malware discovery based on static features. In their approach, Schultz et al. used program executable, byte n-gram & string as a feature vector. The classifier which is implemented by the author is Ripper, Naïve Bayes, and Multi Naïve Bayes to train also test the dataset [1].

Zhu et al. [30] proposed a profoundly proficient approach to extricate authorizations and sensitive APIs as feature and utilized Random Forest to organize the demonstrate to acknowledge between malicious and benign applications. The authors utilized 2130 tests and achieved 88.26% detection correctness with 88.40% sensitivity at the accuracy of 88.16% [4].

V. PROPOSED METHODOLOGY

In this section, author explains the system-learning-based method for detecting fraudulent Android packages. Figure 1 depicts the geometry of our proposed technique, which includes the following advancements:

- Collecting the data set
- Extracting the feature (Opcodes)
- Classification of malware and benign application

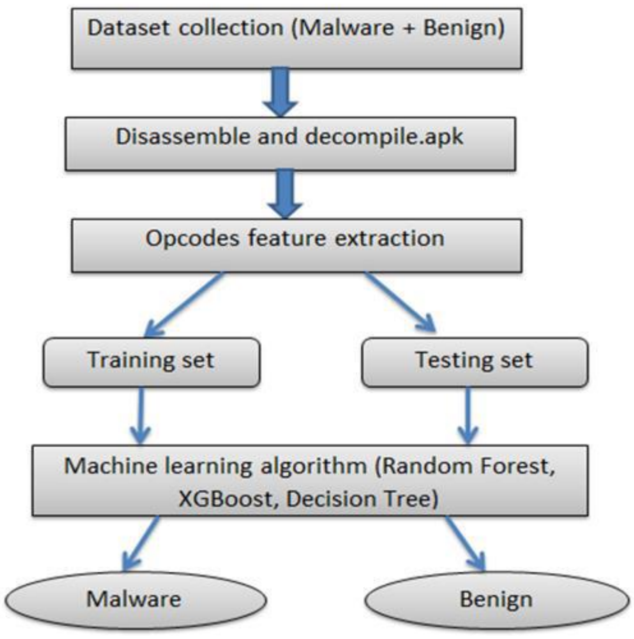
Dataset collection (Malware + Benign)
Disassemble and decompile.apk
Opcodes feature extraction
Training set
Testing set
Machine learning algorithm (Random Forest, XGBoost, Decision Tree)
Malware
Benign

Figure 1: Flow chart of the approach

5.1 Dataset Collection

Authors gather dataset (malicious and benign apks) for Android applications from AndroZoo, which is a creating vault of Android applications. AndroZoo covers the applications which are gathered from the different bases, containing the Google play store commercial center. The dataset is used for the investigation covers 15000 malware and 15000 generous Android application bundle APKs. Author additionally checks the Secure Hash Algorithm (SHA) worth of the applications to guarantee the novel example for investigation [1].

ISSN (online): 2581-3048
Volume 5, Issue 7, pp 56-61, July-2021
https://doi.org/10.47001/IRJIET/2021.507010

5.2 Feature Extraction

In this stage, author extricates the highlights from the dataset that have gathered. In this approach author played out the static investigation of android applications to recognize malevolent applications. For the extraction of highlights, author use Apktool and Androguard [1] figuring out devices and concentrate opcodes from the dataset. During the underlying phase of highlight extraction, author separates an enormous number of highlights in every class. Then, at that point first channel the highlights with the top recurrence of event in the dataset. Figure 2 shows the main 20 frequency feature includes in our dataset [1].

Opcodes: Opcodes assume a fundamental part in the execution of the application. During the writing audit, author track down that in the static investigation, functional codes are fundamental structure blocks during the execution of uses [1].

An opcode distinguishes which fundamental PC activity in the guidance set is to be performed. It advises the PC to accomplish something. Each machine language guidance ordinarily has both an opcode and operands. The opcode resembles an action word in a sentence, and the operands resemble the subject in a sentence.

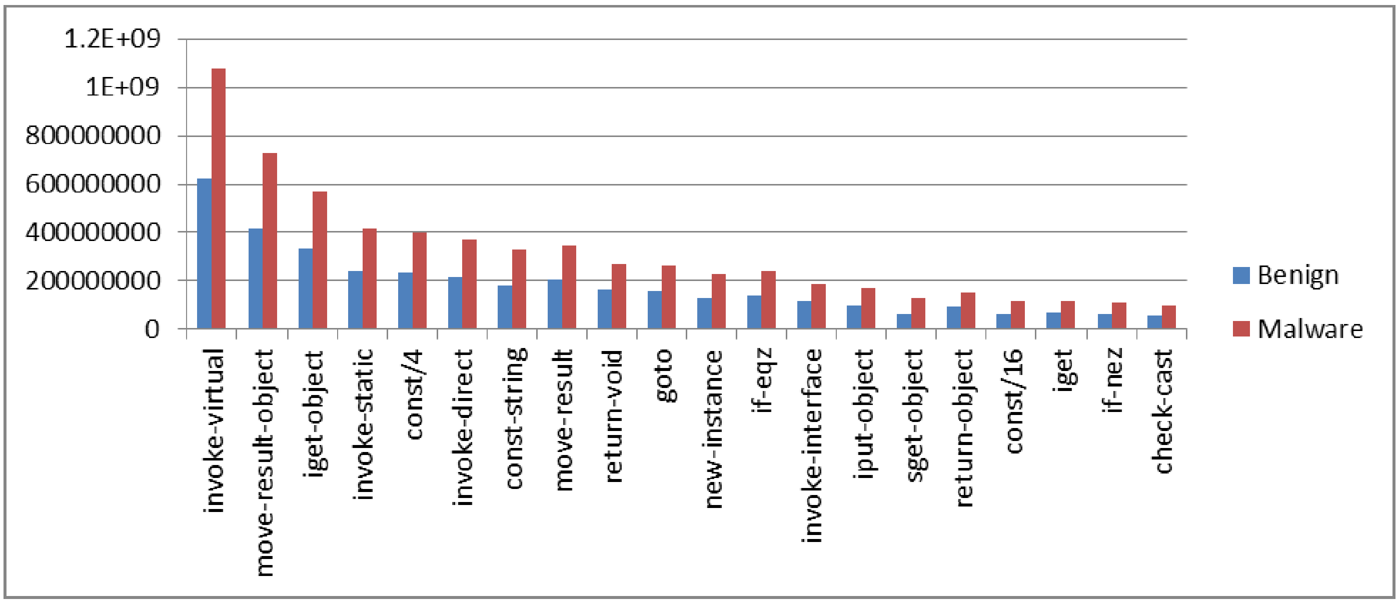

Figure 2: Comparison graph of top 20 opcodes in malware and benign

5.3 Classification

For the classification, we use 4 assorted supervised machine learning classifiers, specifically Random forest, eXtreme Gradient Boosting (XGBoost).

Random forest

A random forest is a supervised machine learning algorithm for classification. It creates a decision tree on data sample and gets the prediction from each of them and finally select best of them by means of voting. It's anything but a stowing strategy that takes perceptions in an arbitrary way and chooses all sections which are unequipped for addressing huge factors at the root for all choice trees.

Table 1: Random forest confusion metrics

	Actual Class	
Predictive Class	21	5
	5	186

XGBoost

XGBoost is a streamlined administered inclination boosting library intended to be somewhat proficient, bendy and convenient. It carries out framework dominating calculations underneath the Gradient Boosting structure. XGBoost manages the cost of an equal tree boosting (also called GBDT, GBM) that cure numerous records mechanical skill issues in a fast and right manner [8].

Table 2: XGBoost confusion metrics

	Actual Class	
Predictive Class	20	7
	10	180

Decision Tree

Decision tree is the greatest influential and renowned gadget in lieu of type and forecast. A Decision tree is a

ISSN (online): 2581-3048

Volume 5, Issue 7, pp 56-61, July-2021

https://doi.org/10.47001/IRJIET/2021.507010

flowchart like tree structure, in which every inner hub means a test on a characteristic, each division addresses an absolute last impact of the test, and each leaf hub (terminal hub) holds a class name.

Table 3: Decision tree confusion metrics

	Actual Class	
Predictive Class	18	7
	9	183

The data set is split into 70%-30% ratio for training and testing. "Author used 10 fold cross validation to validate the performance of the model. In the basic approach, called k-fold cross validation", the training set is parted into k more modest sets the ensuing strategy is followed for everything about k "folds" [2]:

- A model is prepared utilizing of the folds as preparing information;
- The subsequent model is approved on the excess a piece of the data (i.e., it's utilized as a test set to process an exhibition measure like precision) [2].

VI. RESULT AND ANALYSIS

6.1 Measurement Metrics

Accuracy (AC) is the proportion of the total number of corrected predictions. Overall, how often is the classifier correct?

$$\boldsymbol{Accuracy}\,\% = (\boldsymbol{TP}+\boldsymbol{TN})/(\boldsymbol{TM}+\mathrm{T}\boldsymbol{B})\times\mathbf{100}$$

Table 4: confusion metrics

		Actual Class	
		Positive	Negative
Predictive Class	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

6.2 Performance Results

As a final point, author perceive that the ensemble based learning algorithms (e.g. Random Forest, XGBoost, Decision Tree) performs better compared with others learning algorithms.

Table 5: Performance result

Features/Classifiers	Random Forest	XGBoost	Decision Tree
Opcodes (218)	95.39%	92.16%	92.6%

VII. COMPARING WITH EXISTING WORK

Figure [3] show the comparison of accuracy obtained by proposed approach with the existing literature Sun et al. [15], Huang et al. [5], Cai et al. [42], and Tehri et al. [34]. The results show that the proposed approach shows best results as compared to the existing approaches.

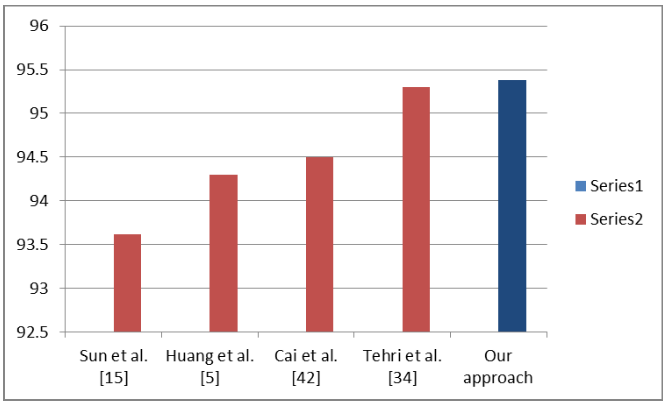

Figure 3: Comparison graph with existing approaches

VIII. CONCLUSIONS

The approach is based on detection of Android applications by identifying the most relevant category of opcodes to discriminate the malicious and benign application. There are three different classification algorithm (Random Forest, XGBoost, Decision tree) are used and achieved 95.3% accuracy with random forest. The methodology has the potential to discover new anomalous applications. This approach is based on static feature.

IX. FUTURE SCOPE

As future work, author examination place is consolidating static and dynamic appraisal in which particular machine learning classifiers are utilized to break down regularly source code similarly as extraordinary component of uses in run time climate [1].

ACKNOWLEDGEMENT

"The author/s acknowledges the Summer Faculty Research Fellow (SFRF-2021) Programme of CEP, IIT Delhi, which enabled Ms. Sonal Pandey to pursue research in IIT Delhi."

REFERENCES

[1] Q. Heal, "QUARTERLY THREAT REPORT Q2-2020," 2020.

[2] Neil, "An Overview of the Android Architecture." Available: https://www.techotopia.com/index.php/An_Overview_

ISSN (online): 2581-3048
Volume 5, Issue 7, pp 56-61, July-2021
https://doi.org/10.47001/IRJIET/2021.507010

of_the_Android_Architecture . [Accessed: 15-Sept-2020]

[3] P. Szor, "The Art of Computer Virus Research and Defense," Symantec Press Publisher, vol. 43, no. 03, pp. 180-200, 2005.

[4] X. Ge, Y. Pan, Y. Fan, and C. Fang, "AMDroid: Android Malware Detection Using Function Call Graphs," Proc. - Companion 19th IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS-C 2019, pp. 71–77, 2019.

[5] N. Huang, M. Xu, N. Zheng, T. Qiao, and K. K. R. Choo, "Deep android malware classification with API-based feature graph," Proc. - 2019 18th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. IEEE Int. Conf. Big Data Sci. Eng. Trust. 2019, pp. 296–303, 2019.

[6] Z. Zhang, C. Chang, P. Han, and H. Zhang, "Packed malware variants detection using deep belief networks," MATEC Web Conf., vol. 309, p. 02002, 2020.

[7] J. Hernandez Jimenez and K. Goseva-Popstojanova, "Malware Detection Using Power Consumption and Network Traffic Data," Proc. - 2019 2nd Int. Conf. Data Intell. Secur. ICDIS 2019, pp. 53–59, 2019.

[8] Y. Zhang, Q. Huang, X. Ma, Z. Yang, and J. Jiang, "Using multi-features and ensemble learning method for imbalanced Malware classification," Proc. - 15th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. 10th IEEE Int. Conf. Big Data Sci. Eng. 14th IEEE Int. Symp. Parallel Distrib. Proce, pp. 965–973, 2016.

[9] A.Govindaraju, "Exhaustive Statistical Analysis for Detection of Metamorphic Malware," 2010.

[10] H. Florian, "Introduction to Malware Analysis Techniques," 2015.

[11] J.-Y. Xu, a. H. Sung, P. Chavez, and S. Mukkamala, "Polymorphic malicious executable scanner by API sequence analysis," Fourth Int. Conf. Hybrid Intell. Syst., pp. 0–5, 2004.

[12] A.Sharma and S. K. Sahay, "An effective approach for classification of advanced malware with high accuracy," Int. J. Secur. its Appl., vol. 10, no. 4, pp. 249–266, 2016.

[13] S. K. Sharma, Sanjay and Krishna, C Rama and Sahay, "Detection of advanced malware by machine learning techniques," in Soft Computing: Theories and Applications, 2019, pp. 333–342.

[14] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," Soft Comput., vol. 20, no. 1, pp. 343–357, 2016.

[15] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant Permission Identification for Machine-Learning-Based Android Malware Detection," IEEE Trans. Ind. Informatics, vol. 14, no. 7, pp. 3216–3225, 2018.

[16] Jyoti Landage, M. P. Wankhade, "Malware and Malware Detection Techniques: A Survey," International Journal of Engineering Research & Technology (IJERT), vol. 2, Issue 12, pp. 61–68, 2018.

[17] R. H. D. Ke Xu, Yingjiu Li, "Iccdetector: Icc-based malware detection on android," in Information Forensics and Security, 2016, pp. 1252–1264.

[18] K. Wain and Y. Au, "by A thesis submitted in conformity with the requirements Graduate Department of Electrical and Computer Engineering c Copyright 2012 by Kathy Wain Yee Au," 2012.

[19] G. Tao, Z. Zheng, Z. Guo, and M. R. Lyu, "MalPat: Mining Patterns of Malicious and Benign Android Apps via Permission-Related APIs," IEEE Trans. Reliab., vol. 67, no. 1, pp. 355–369, 2018.

[20] M. C. Sanjeev Das, Yang Liu, Wei Zhang, "Semantics-based online malware detection: Towards efficient real-time pro- tection against malware," in Information Forensics and Security, 2016, pp. 289–302.

[21] A.Sharma and S. K. Sahay, "Evolution and Detection of Polymorphic and Metamorphic Malwares: A Survey," Int. J. Comput. Appl., vol. 90, no. 2, pp. 7–11, 2014.

[22] K. Griffin, S. Schneider, X. Hu, and T. C. Chiueh, "Automatic generation of string signatures for malware detection," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2009, vol. 5758 LNCS, pp. 101–120.

[23] I.A. Saeed, A. Selamat, and A. M. A. Abuagoub, "A Survey on Malware and Malware Detection Systems," vol. 67, no. 16, pp. 25–31, 2013.

[24] A.Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey," Inf. Secur. Tech. Rep., vol. 14, no. 1, pp. 16–29, 2009.

[25] M. G. Schultz, E. Eskin, and S. J. Stolfo, "Data Mining Methods for Detection of New Malicious Executables," 2001.

[26] D. Bilar, "Opcodes As Predictor for Malware," Int. J. Electron. Secur. Digit. Forensic, vol. 1, no. 2, pp. 156–168, 2007.

[27] K. Allix, T. F. Bissyandé, Q. Jérome, J. Klein, R. State, and Y. Le Traon, "Large-scale machine learning-based malware detection," in Proceedings of the 4th ACM conference on Data and application security and privacy - CODASPY '14, 2014, pp. 163–166.

[28] C. Wang, Z. Qin, J. Zhang, and H. Yin, "A malware variants detection methodology with an opcode based

feature method and a fast density based clustering algorithm," pp. 481–487, 2016.

[29] E. B. Bahman Rashidi, Carol Fung, "Android resource usage risk assessment using hidden Markov model and online learning," in Computers & Security, 2017, pp. 90–107.

[30] H. J. Zhu, Z. H. You, Z. X. Zhu, W. L. Shi, X. Chen, and L. Cheng, "DroidDet: Effective and robust detection of android malware using static analysis along with rotation forest model," Neurocomputing, vol. 272, pp. 638–646, 2018.

[31] A.Sharma and S. K. Sahay, "An investigation of the classifiers to detect android malicious apps," 2016.

[32] D. Ö. Şahin, O. E. Kural, S. Akleylek, and E. Kiliç, "New results on permission based static analysis for Android malware," 6th Int. Symp. Digit. Forensic Secur. ISDFS 2018 - Proceeding, vol. 2018-Janua, pp. 1–4, 2018.

[33] J. Rudy, "Adapting Text Categorization for Manifest based Android Malware Detection," Computer Science Journal, vol. 19, no. 3, pp. 257–279, 2018.

[34] L. Taheri, A. F. A. Kadir, and A. H. Lashkari, "Extensible android malware detection and family classification using network-flows and API-calls," Proc. - Int. Carnahan Conf. Secur. Technol., vol. 2019-October, no. Cic, 2019.

[35] M. Kruczkowski and E. Niewiadomska-Szynkiewicz, "Comparative study of supervised learning methods for 6 malware analysis," J. Telecommun. Inf. Technol., vol. 2014, no. 4, pp. 24–33, 2014.

[36] I.Firdausi, C. Lim, A. Erwin, and A. S. Nugroho, "Analysis of machine learning techniques used in behavior-based malware detection," Proc. - 2010 2nd Int. Conf. Adv. Comput. Control Telecommun. Technol. ACT 2010, pp. 201–203, 2010.

[37] N. Milosevic, A. Dehghantanha, and K. K. R. Choo, "Machine learning aided Android malware classification," Comput. Electr. Eng., vol. 61, pp. 266–274, 2017.

[38] Ke Xu, Yingjiu Li, Robert H. Deng "ICC Detector: ICC Based Malware Detection on Android," IEEE Transactions on Information Forensics and Security, vol: 11, Issue: 6, pp. 1252–1264, 2016.

[39] Neha Tarar, Shweta Sharma, Dr. C. Rama Krishna "Analysis and Classification of Android Malware using Machine Learning Algorithms," IEEE 3rd international conference on Inventive Computation Technologies, vol: 10, Issue: 3, 2018.

[40] Andrea Saracino, Daniele Sgandurra, Gianluca Dini and Fabio Martinelli "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," IEEE Transactions on Dependable and Secure Computing, vol: 15, pp. 83 - 97 2018.

[41] Sonal Pandey, C. Rama Krishna, Ashu Sharma, Sanjay Sharma "Detection of Android Malware Using Machine Learning Techniques," Innovations in Computer Science and Engineering, vol: 171, pp. 663 - 675 2021.

[42] Haipeng Cai, Na Meng, Barbara Ryder, Daphne Yao "DroidCat: Effective Android Malware Detection and Categorization via App-Level Profiling," IEEE Transactions on Information Forensics and Security, vol: 14, pp. 1455 - 1470 2015.

Citation of this Article:

Sonal Pandey, Ram Lal, "Opcode-Based Android Malware Detection Using Machine Learning Techniques" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 5, Issue 7, pp 56-61, July 2021. Article DOI https://doi.org/10.47001/IRJIET/2021.507010

****