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

ISSN (online): 2581-3048

Volume 8, Issue 3, pp 206-211, March-2024

https://doi.org/10.47001/IRJIET/2024.803029

Permission Based Android Malware Detection Using Machine Learning

[1]Sonal Pandey, [2]Satyasheel

[1]DBRAU Agra, India. E-mail: pandey.sonal88@gmail.com
[2]ADRDE DRDO, India. E-mail: satyasheelforyou@gmail.com

***Abstract* - Due to the open architecture of the Android operating system, there has also been a huge increase in mobile malware. With the growth in amount, variants, diversity and sophistication in malware, conventional methods often fail to detect malicious applications. Signatures based technologies work efficient for known malware but fail to detect unknown or new malware. In this paper author will appliance an approach to detect the unfamiliar Android malware using machine learning techniques. In our approach, we extract permissions (AOSP and third party permissions) features for getting high accuracy. Then features were selected along with separate apks (malware and benign files) in training and testing classifiers. We evaluate our method on AndroZoo dataset (15000 malware and 15000 benign Apks) We use Random forest classifiers for classification of Android malware and achieved 91.1% accuracy with AOSP and 72.3% accuracy with Third Party Permission.**

Keywords: Malware, Metamorphic malware, Android, Machine Learning, Random forest, Decision Tree, XGBoost.

I. INTRODUCTION

Malware is malicious software that enters your computer Perform unnecessary tasks while compromised computer system security policy in data terms Confidentiality, Integrity and Availability. Malware has the ability to do so modify or remove software packages from your system. The direction of intentionally destroying the essence of the system function. Examples of malware are Trojan horses, spyware, adware, worms.

II. OBJECTIVE

- To study the existing malware detection techniques.
- To extract prominent features (requested AOSP permissions, requested third-party permissions.) from executable of the allocated dataset by performing static analysis.
- To adopt a Random forest, Decision Tree, XGBoost machine learning model for the classification of malware.

III. LITERATURE REVIEW

Allix et al. [18] proposed a novel approach in 2014 to extract the control flow graph from the application program that is a more meaningful way than n-gram representation. The authors used a sizeable dataset (over 50000) of android application and implemented using machine learning classifiers viz. Random Forest, J48, LibSVM and JRip using 10-fold cross-validation.

Sanjeev Das et al. [25] used field-programmable gate arrays and processors in their proposed hardware-enhanced architecture i.e. GuardOL. Authors used system call as the feature to detect malicious apps. The importance of the author's design was that the approach in the first 30% of the execution recognizes 46% of the malware and after 100% of execution, 97% of the samples have been identified with 3% FP.

Bahman Rashidi et al. [26] proposed an android resources usage risk assessment called XDroid. In real-time their model can inform users about the risk level of the application, and can dynamically update the parameters of the model by the client's preferences and from the on-line algorithm. They have used the Drebin malware dataset and demonstrated that their approach could estimate the risk levels of the malware up to 82% accuracy and with the user input it can provide an adaptive risk assessment [1].

Recently, Sharma and Sahay et al. [28] examined the five classifiers on the Drebin dataset using the opcodes occurrence as a feature and got an accuracy of 79.27% by functional tree classifier for malicious application detection.

Sahin et al. [29] proposed a permission-based Android malware system to detect malicious applications. Unlike other studies, the authors proposed a permission weight approach. Each of the permissions is given a different score using this approach. Then, K-nearest Neighbor (KNN) and Naïve Bayes (NB) algorithms are applied and got 90.76% accuracy. According to the Authors, the proposed approach has better results than the other ones.

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

ISSN (online): 2581-3048

Volume 8, Issue 3, pp 206-211, March-2024

https://doi.org/10.47001/IRJIET/2024.803029

Coban et al. [30] proposed a static malware detection system by using text categorization techniques. Authors applied text mining techniques like feature extraction by using bag-of-words, n-grams, etc. from manifest content of suspicious programs, then apply text classification methods to detect malware. Their approach is capable of detecting malicious applications with an accuracy between 94.0% and 99.3%.

Xiuting et al. [31] proposed AMDroid that uses function call graphs (FCGs) representing the behaviors of applications and applies graph kernels to learn the structural semantics of applications from FCGs automatically. The authors evaluated AMDroid on the Genome Project, and the experimental results show that AMDroid is effective in identifying Android malicious applications with 97.49% detection accuracy. Taheri et al. [32] proposed the second part of their CICAndMal2017 dataset publicly accessible which incorporates permissions and intents as static features, and API calls as dynamic features. The authors examined these features with our two-layer Android malware analyzer. Consistent with their approach, they succeeded in achieving 95.3% precision in Static-Based Malware Binary Classification at the primary layer, 83.3% precision in Dynamic-Based Malware Category Classification and 59.7% precision in Dynamic-Based Malware Family Classification at the second layer. Huang et al. [33] focused on the application programming interface as features and proposed methods to detect Android malicious applications. First, the Authors proposed a selection method for API features associated with the malware class. Second, they further explored structure relationships between these APIs and map to a matrix interpreted because the hand refined API-based feature graph. Third, a CNN-based classifier is trained for the API-based feature graph classification. Authors used a dataset containing 3,697 malware applications and 3,312 benign apps demonstrate that the chosen API feature is effective for Android malware classification, just the highest 20 APIs got 94.3% detection accuracy under Random Forest classifier.

Neha et al. [41] focused on different machine learning based analysis methods used for the classification of Android malware applications. In this paper, Opcode-based Android malware analysis approach has been proposed and applied different machine learning algorithm (Naïve Bayes, AdaBoost, Random Tree, Bagging & SMO) then achieved 99.5% accuracy with a 0.995 TPR.

IV. PROPOSED METHODOLOGY

Author defines the methodology for detecting fraudulent Android packages. Figure 1 explain the proposed technique, which includes the following advancements:

- Collecting the data set
- Extracting the permissions
- Classification of malware and benign application

4.1 Dataset Collection

We collect malicious and benign Android applications from AndroZoo [6], which is a growing repository of Android apps. AndroZoo contains the applications that are collected from the various sources, including the Google play store marketplace. The dataset we use for the analysis contains 15000 malware and 15000 benign APKs. We also check the Secure Hash Algorithm (SHA) value of the applications to ensure the unique sample for analysis.

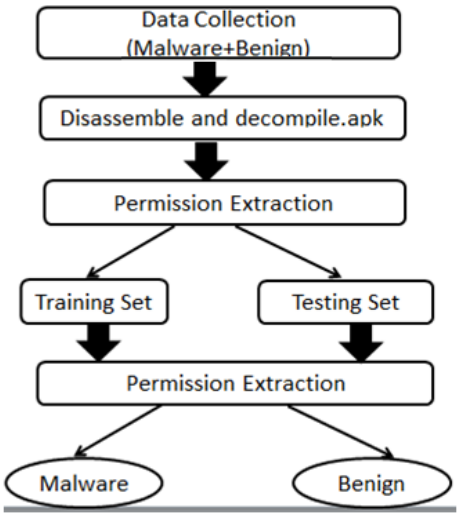

Figure 1: Flow chart of the approach

4.2 Feature Extraction

In this phase, we extract the features from the dataset that we have collected. For the extraction of features, we use Apktool and Androguard [1] reverse engineering tools. We extract seven categories of features, namely requested Android Open Source Project (AOSP) permissions, requested third-party permissions. During the initial stage of feature extraction, we extract a huge number of features. Then we first filter the features with the top frequency of occurrence in the dataset.

- **Permissions:**

Permissions are used to protect the privacy of an Android user, and a few applications also need permission to access users' sensitive data like SMS, contact, etc. Some applications also request third part permission which is not mentioned in the android open source project. The combination of permission sometimes reflects the malicious behavior. Therefore, we extract two types of permission as features AOSP and third party permission.

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

ISSN (online): 2581-3048

Volume 8, Issue 3, pp 206-211, March-2024

https://doi.org/10.47001/IRJIET/2024.803029

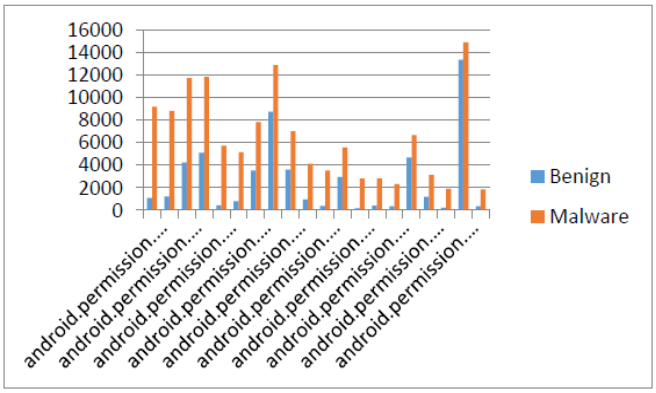

Figure 2: Comparison Graph of Top 20 AOSP Permission in Malware and Benign

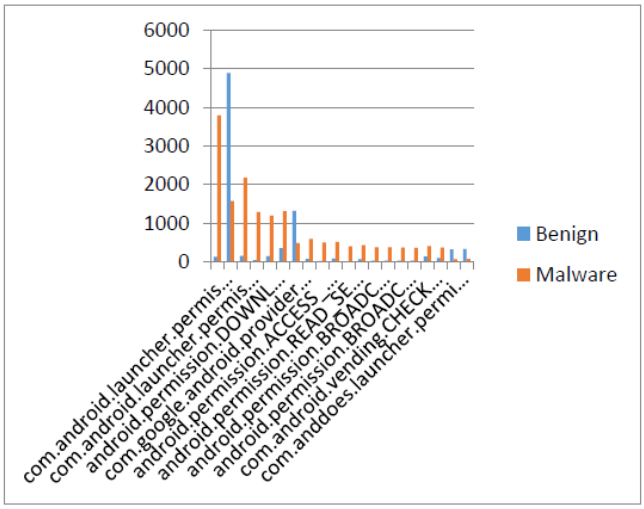

Figure 3: Comparison Graph of Top 20 Third Party Permission in Malware and Benign

4.3 Classification

This section discusses the machine learning classifiers which is used in our work. For the classification, we use Random forest supervised machine learning classifiers.

• 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 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].

V. RESULT AND ANALYSIS

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

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

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

ISSN (online): 2581-3048

Volume 8, Issue 3, pp 206-211, March-2024

https://doi.org/10.47001/IRJIET/2024.803029

Table 4: Confusion metrics

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

5.1 Performance Results

As a final point, author perceives the result based on Random Forest, XGBoost, Decision Tree classifier.

Table 5: Performance result

Features/classifier	Random Forest	XGBoost	Decision Tree
AOSP (206)	0.911312	0.878179	0.899933
Third party permissions (8939)	0.723896	0. 72423	0.725904

VI. CONCLUSIONS

The approach is based on detection of Android applications by identifying the most relevant category of permissions to discriminate the malicious and benign application. Random forest, XGBoost, Decision Tree classification algorithm is used and achieved 91.1% accuracy with AOSP and 72.3% accuracy with Third Party Permission through Random forest . The methodology has the potential to discover new anomalous applications. This approach is based on static feature.

REFERENCES

[1] IDC, "Smartphone market share," 2019. [Online]. Available: ttps://www.idc.com/promo/%0Asmartphone-market-share/os. [Accessed: 02-Aug-2019].

[2] E. Protalinski, "Android passes 2.5 billion monthly active devices Venture beat.," 2019. [Online]. Available: https://venturebeat.com/2019/05/07/%0Aandroid-passes-2-5-billion-monthly-active-devices/. [Accessed: 15-Sep-2019].

[3] Q. Heal, "QUARTERLY THREAT REPORT Q2-2019," 2019.

[4] Android, "Platform Architecture Agenda," Architecture, 2018. [Online]. Available: https://developer.android.com/guide/%0Aplatform. [Accessed: 22-Sep-2019].

[5] Neil, "An Overview of the Android Architecture." Available: https://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture. [Accessed: 15-Sept-2019]

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

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

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

ISSN (online): 2581-3048
Volume 8, Issue 3, pp 206-211, March-2024
https://doi.org/10.47001/IRJIET/2024.803029

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

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

[21] 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.

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

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

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

[29] 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.

[30] J. Rudy, "Early rd Early bi rd," vol. 19, no. 3, pp. 257–279, 2018.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

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

[40] 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.

[41] 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.

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

ISSN (online): 2581-3048

Volume 8, Issue 3, pp 206-211, March-2024

https://doi.org/10.47001/IRJIET/2024.803029

Citation of this Article:

Sonal Pandey, Satyasheel, "Permission Based Android Malware Detection Using Machine Learning", Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 8, Issue 3, pp 206-211, March 2024. Article DOI https://doi.org/10.47001/IRJIET/2024.803029

****

