Applied GIS

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

Malware Detection A Framework for Reverse Engineered Android Applications through Machine Learning Algorithms

Dr.Putta.Srivani[1], Ch.Vandana[2], P.Vinusha[3], S.Sathvika[4],A.Manimanognya[5]
[1]Professor, Department of CSE(CS), MallaReddy Engineering College for Women, Hyderabad, TS, India.
Email: pulla.srivani@gmail.com
[2,3,4,5]UG Students, Department of CSE(CS), MallaReddy Engineering College for Women, Hyderabad, TS, India.

ABSTRACT

Today, Android is one of the most used operating systems in smartphone technology. This is the main reason, Android has become the favorite target for hackers and attackers. Malicious codes are being embedded in Android applications in such a sophisticated manner that detecting and identifying an application as a malware has become the toughest job for security providers. In terms of ingenuity and cognition, Android malware has progressed to the point where they're more impervious to conventional detection techniques. Approaches based on machine learning have emerged as a much more effective way to tackle the intricacy and originality of developing Android threats. They function by first identifying current patterns of malware activity and then using this information to distinguish between identified threats and unidentified threats with unknown behavior. This research paper uses Reverse Engineered Android applications' features and Machine Learning algorithms to find vulnerabilities present in Smartphone applications. Our contribution is twofold. Firstly, we propose a model that incorporates more innovative static feature sets with the largest current datasets of malware samples than conventional methods. Secondly, we have used ensemble learning with machine learning algorithms such as AdaBoost, SVM, etc. to improve our model's performance. Our experimental results and findings

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

exhibit 96.24% accuracy to detect extracted malware from Android applications, with a 0.3 False Positive Rate (FPR). The proposed model incorporates ignored detrimental

INTRODUCTION

To this degree, it is guaranteed that mobile devices are an integral part of most people's daily lives. Furthermore, Android now controls the vast majority of mobile devices, with Android devices accounting for an average of 80% of the global market share over the past years. With the ongoing plan of Android to a growing range of smart phones and consumers around the world, malware targeting Android devices has increased as well. Since it is an open-source operating system, the level of danger it poses, with malware authors and programmers implementing unwanted permissions, features and application components in Android apps. The option to expand its capabilities with third-party software is also appealing, but this capability comes with the risk of malicious device attacks. When the number of smart phone apps

features such as permissions, intents, API calls, and so on, trained by feeding a solitary arbitrary feature, extracted by reverse engineering as an input to the machine.

increases, so does the security problem with unnecessary access to different personal resources. As a result, the applications are becoming more insecure, and they are stealing personal information, SMS frauds, ransom ware, etc.

In contrast to static analysis methods such as a manual assessment of AndroidManifest.xml, source files and Dalvik Byte Code and the complex analysis of a managed environment to study the way it treats a program, Machine Learning includes learning the fundamental rules and habits of the positive and malicious settings of apps and then data-venabling. The static attributes derived from an application are extensively used in machine learning methodologies and the tedious task of this can be relieved if the static features of reverse-engineered Android

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

Applications are extracted and use machine learning SVM algorithm, logistic progression, ensemble learning and other algorithms to help train the model for prediction of these malware applications [1].

Machine learning employs a range of methodologies for data classification. SVM (Support Vector Machine) is a strong learner that plots each data item as a point in n-dimensional space (where n denotes the number of features you have), with the value of each feature becoming the vector value. Then it executes classification by locating the hyper-plane that best distinguishes the two groups, leading to an improvement identification property for any two parameters. Conversely, boosting or ensemble techniques like Adaboost are assigned higher weights to rectify the behavior of misclassified variables in conjunction with other machine algorithms. When combined alongside weak classifiers, our preliminary model benefits from deploying such models since they have a high degree of precision or classification. [2], [3], [4], supports classifiers in their system models to find the highest accuracy. Although using ensemble or strong classifiers can cause problems like multi collinearity, which in a regression model, occurs when two or more independent variables are strongly associated with one another. In multivariate regression, this indicates that one regression analysis may be forecasted from another independent variable. This scope of the study can be presented as a detection journal analysis itself and can present several experimentations and results based on machine learning models [5], [6].

When an app has access to a resource in the most recent versions of Android OS, it must ask the OS for approval, and the OS will ask the user if they wish to grant or refuse the request via a pop-up menu. Many reports have been performed on the success of this resource management approach. The studies showed consumers made decisions by giving all requested access

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

to the applications to their privileges requests [7]. In contrast to this, over 70% of Android mobile applications seek extra access that is not needed. They also sought a permit that is not needed for the app to run. A chess game that asks for photographs or requests for SMS and phone call permits, or loads unwanted packages are an example of an extra requested authorization. So, trying to assess an app's vindictiveness and not understanding the app is a tough challenge. As a result, successful malicious app monitoring will provide extra information to customers to assist them and defend them from information disclosure [8]. Figure 1 elaborates the android risk framework through the Google Play platform, which is then manually configured by the android device developers.

Contrary to other smart phone formats, such as IOS, Android requires users to access apps from untrusted outlets like file- sharing sites or third-party app stores. The malware virus problem has become so severe that 97 % of all Smartphone malware now targets Android phones. In a year, approximately 3.25 million new malware Android applications are discovered as the growth of smartphones increases. This loosely amounts to a new malware android version being introduced every few seconds [9]. The primary aim of mobile malware is to gain entrance to user data saved on the computer and user information used in confidential financial activities, such as banking. Infected file extensions, files received via Bluetooth, links to infected code in SMS, and MMS application links are all ways that mobile malware can propagate [10]. There are some strategies for locating apps that need additional features. Hopefully, by using these techniques, it would be possible to determine whether the applications that were flagged as questionable and needed additional authorization are malicious.

Static analysis methodologies are the most fundamental of all approaches. Until operating programs, the permissions and source codes are

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

examined [11]. For many machine learning tasks, such as enhancing predictive performance or simplifying complicated learning problems, ensemble learning is regarded as the most advanced method. It enhances a single model's prediction performance by training several models and combining their predictions. Boosting, bagging, and random forest are examples of common ensemble learning techniques [12]. In summary, the main contributions of our study are as follows:

1) We present a novel subset of features for static detection of Android malware, which consists of seven additional selected feature sets that are using around 56000 features from these categories. On a collection of more than 500k benign and malicious Android applications and the highest malware sample set than any state-of-the-art approach, we assess their stability. The results obtain a detection increase in accuracy to 96.24 % with 0.3% false-positives.

2) With the additional features, we have trained six classifier models or machine learning algorithms and also implemented a Boosting ensemble learning approach (AdaBoost) with a Decision Tree based on the binary classification to enhance our prediction rate.

3) Our model is trained on the latest and large time aware samples of malware collected within recent years including the latest Android API level than state-of-the-art approaches. This research paper incorporates binary vector mapping for classification by allocating 0 to malicious applications and 1 for non-harmful and for predictive analysis of each application fed to the model implemented in the study. The technique eases the process by reducing fault predictive errors. Figure 2 shows the procedure for a better understanding of the concept applied later in our study. The paper passes both the categories of applications through static analysis and then is further processed for feature

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

extraction. We presented features in 0's and 1's after extraction. Matrix displays the extraction characteristics of each application used in the dataset. There are major issues to be addressed to incorporate our strategy. High measurements of the features will make it difficult to identify malware in many real-world Android applications. Certain features overlap with innocuous apps and malware [13]. In comparison, the vast number of features will cause high throughput computing. Therefore, we can learn from the features directly derived from Android apps, the most popular and significant features. The paper implements prediction models and various computer ensemble teaching strategies to boost and enhance accuracy to resolve this problem [14]. Feature selection is an essential step in all machine-based learning approaches. The optimum collection of features will not only help boost the outcomes of tests but will also help to reduce the compass of most machine-based learning algorithms [15]. Studies have extensively suggested three separate methods for identifying android malware: static, interactive meaning dynamically, and synthetic or hybrid. Static analysis techniques look at the code without ever running it, so they're a little sluggish if carried out manually and have to face a lot of false positives [16]. Data obfuscation and complex code loading are both significant pitfalls of the technique. That is why automated operation helps to achieve reliability, accuracy, and lesser time utilization [17]. Reverse engineer Android applications and extract features and do static analysis from them without having to execute them. This method entails examining the contents of two files: AndroidManifest.xml and classes.dex, and working on the file with the .apk extension. Feature selection techniques and classification algorithms are two crucial areas of feature- based types of fraudulent applications. Feature filtering methods are used to reduce the dimension size of a dataset. Any of the functions (attributes) that aren't helpful in the study are omitted from the data collection because of this. The

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

remaining features are chosen by weighing the representational strength of all the dataset's features [18]. Parsing tools can help learn which permissions, packages or services an application offers by analyzing the AndroidManifest.xml file, such as permission android.permission.call phone, which allows an application to misuse calling abilities. The paper elaborates exactly what sort of sensitive API the authors could name by decoding the classes.dex file with the Jadx-gui disassembler [19]. In certain cases, including two permissions in a single app can signify the app's possible malicious attacks. For example, an application with RECEIVE SMS and WRITE SMS permissions can mask or interfere with receiving text messages [20] or applying sensitive API such as sendTextMessage() can also be harmful and lead to fraud and stealing. Until we started our main idea of the project. The fact explained that Android applications pose a lot of threats to its user because of the unnecessary programs compiled inside them and explained why it is necessary to automate the process of static analysis for the efficient detection of malware applications based on the extracted features. The rest of the paper is planned as follows. Related works are examined in Section II. Section III will present the design and method of our model. Section IV elaborates the assessment findings and future threats. The experiments and results will be dilated and performed in Sections V and VI. Section VII includes our research issues, recommendations, and conclusions for the future.

EXISTING SYSTEM

The methods proposed in this related work contribute to key aspects and a

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

higher predictive rate for malware detection. Certain research has focused on increasing accuracy, while others have focused on providing a larger dataset, some have been implemented by employing various feature sets, and many studies have combined all of these to improve detection rate efficiency. In [21] the authors offer a system for detecting Android malware apps to aid in the organization of the Android Market. The proposed framework aims to provide a machine learning-based malware detection system for Android to detect malware apps and improve phone users' safety and privacy. This system monitors different permission-based characteristics and events acquired from Android apps and examines these features employing machine learning classifiers to determine if the program is goodware or malicious.

The paper uses two datasets with collectively 700 malware samples and 160 features. Both datasets achieved approximately 91% accuracy with Random Forest (RF) Algorithm. [22] Examines 5,560 malware samples, detecting 94 % of the malware with minimal false alarms, where the reasons supplied for each detection disclose key features of the identified malware. Another technique [23] exceeds both static and dynamic methods that rely on system calls in terms of resilience. Researchers demonstrated the consistency of the model in attaining maximum classification performance and better accuracy compared to two state-of-the-art peer methods that represent both static and dynamic methodologies over for nine years through three interrelated assessments with satisfactory malware samples from different sources. Model continuously achieved 97% F1- measure accuracy for identifying applications or categorizing malware.

[24] The authors present a unique Android malware detection approach dubbed Permission- based Malware Detection Systems (PMDS) based on a

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

study of 2950 samples of benign and malicious Android applications. In PMDS, requested permissions are viewed as behavioral markers, and a machine learning model is built on those indicators to detect new potentially dangerous behavior in unknown apps depending on the mix of rights they require. PMDS identifies more than 92–94% of all heretofore unknown malware, with a false positive rate of 1.52–3.93%.

The authors of this article [25] solely use the machine learning ensemble learning method Random Forest supervised classifier on Android feature malware samples with 42 features respectively. Their objective was to assess Random Forest's accuracy in identifying Android application activity as harmful or benign. Dataset 1 is built on 1330 malicious apk samples and 407 benign ones seen by the author. This is based on the collection of feature vectors for each application. Based on an ensemble learning approach, Congyi proposes a concept in [26] for recognizing and distinguishing Android malware.

Disadvantages

- The system is not implemented MACHINE LEARNING ALGORITHM AND ENSEMBLE LEARNING.
- The system is not implemented Reverse Engineered Applications characteristics.

Proposed System

1) We present a novel subset of features for static detection of Android malware, which consists of seven additional selected feature sets that are using around 56000 features from these categories. On a collection of more than 500k benign and malicious Android applications and the highest malware sample set than any state-of-the-art approach, we assess their stability. The results obtain a detection increase in accuracy to 96.24 % with 0.3% false-positives.

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

2) With the additional features, we have trained six classifier models or machine learning algorithms and also implemented a Boosting ensemble learning approach (AdaBoost) with a Decision Tree based on the binary classification to enhance our prediction rate. 3) Our model is trained on the latest and large time aware samples of malware collected within recent years including the latest Android API level than state-ofthe-art approaches.

- The system used in this study also incorporates larger feature sets for classification. Although this problem arises in machine learning quite often to some extent choosing the type of model for detection or classification can highly impact the high dimensionality of the data being used.

Advantages

- The proposed system chooses the characteristics based on their capability to display all data sets. Enhanced efficiency by reducing the dataset size and the hours wasted on the classification process introduces an effective function selection process.

Modules

Service Provider

In this module, the Service Provider has to login by using valid user name and password. After login successful he can do some operations such as

Login, Browse App Data Sets and Train & Test, View Mobile Apps Trained and Tested Accuracy in Bar Chart, View Mobile App Trained and Tested Accuracy Results, View Malware Prediction

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

Type, View Malware Prediction Ratio, Download Predicted Data Sets, View Malware Prediction Ratio Results, View All Remote Users.

View and Authorize Users

In this module, the admin can view the list of users who all registered. In this, the admin can view the user's details such as, user name, email, address and admin authorizes the users.

Remote User

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like REGISTER AND LOGIN, PREDICT MALWARE DETECTION TYPE, VIEW YOUR PROFILE.

Decision tree classifiers

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes C1, C2, …, Ck is as follows:

Step 1. If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf labeled with this class

Step 2. Otherwise, let T be some test with possible outcomes O1, O2,…, On. Each object in S has one outcome for T so the test partitions S into subsets S1, S2,… Sn where each object in Si has outcome Oi for T. T becomes the root of the decision tree and for each outcome Oi we build a subsidiary decision tree by invoking

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

the same procedure recursively on the set Si.

Gradient boosting

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.[1][2] When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest.A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.

Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho[1] using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.).The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman[13] in order to construct a

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

collection of decision trees with controlled variance.

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

SVM

In classification tasks a discriminant machine learning technique aims at finding, based on an *independent and identically distributed* (*iid*) training dataset, a discriminant function that can correctly predict labels for newly acquired instances. Unlike generative machine learning approaches, which require computations of conditional probability distributions, a discriminant classification function takes a data point *x* and assigns it to one of the different classes that are a part of the classification task. Less powerful than generative approaches, which are mostly used when prediction involves outlier detection, discriminant approaches require fewer computational resources and less training data, especially for a multidimensional feature space and when only posterior probabilities are needed. From a geometric perspective, learning a classifier is equivalent to finding the equation for a multidimensional surface that best separates the different classes in the feature space.

SVM is a discriminant technique, and, because it solves the convex optimization problem analytically, it always returns the same optimal hyperplane parameter—in contrast to *genetic algorithms* (*GAs*) or *perceptrons*, both of which are widely used for classification in machine learning. For perceptrons, solutions are highly dependent on the initialization and termination criteria. For a specific kernel that transforms the data from the input space to the feature space, training returns uniquely defined SVM model parameters for a given training set,

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

whereas the perceptron and GA classifier models are different each time training is initialized. The aim of GAs and perceptrons is only to minimize error during training, which will translate into several hyperplanes' meeting this requirement.

Results

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

CONCLUSION

In this research, we devised a framework that can detect malicious Android applications. The proposed technique takes into account various elements of machine learning and achieves a 96.24% in identifying malicious Android applications. We first define and pick functions to capture and analyze Android apps' behavior, leveraging reverse application engineering and AndroGuard to extract features into binary vectors and then use python build modules and split shuffle functions to train the model with benign and malicious datasets. Our experimental findings show that our suggested model has a false positive rate of 0.3 with 96% accuracy in the given environment with an enhanced and larger feature and sample sets. The study also discovered that when dealing with classifications and high-dimensional data, ensemble and strong learner algorithms perform comparatively better. The suggested approach is restricted in terms of static analysis, lacks sustainability concerns, and fails to address a key multi collinearity barrier. In the future, we'll consider model resilience in terms of enhanced and dynamic features. The issue of dependent variables or high inter correlation between machine algorithms before employing them is also a promising field.

REFERENCES

[1] A. O. Christiana, B. A. Gyunka, and A. Noah, “Android Malware Detection through Machine Learning Techniques:

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

A Review," Int. J. Online Biomed. Eng. IJOE, vol. 16, no. 02, p. 14, Feb. 2020, doi: 10.3991/ijoe.v16i02.11549. [2] D. Ghimire and J. Lee, "Geometric Feature-Based Facial Expression Recognition in Image Sequences Using Multi-Class AdaBoost and Support Vector Machines," Sensors, vol. 13, no. 6, pp. 7714–7734, Jun. 2013, doi: 10.3390/s130607714. [3] R. Wang, "AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review," Phys. Procedia, vol. 25, pp. 800–807, 2012, doi: 10.1016/j.phpro.2012.03.160. [4] J. Sun, H. Fujita, P. Chen, and H. Li, "Dynamic financial distress prediction with concept drift based on time weighting combined with Adaboost support vector machine ensemble," Knowl.-Based Syst., vol. 120, pp. 4–14, Mar. 2017, doi: 10.1016/j.knosys.2016.12.019. [5] A. Garg and K. Tai, "Comparison of statistical and machine learning methods in modelling of data with multicollinearity," Int. J. Model. Identif. Control, vol. 18, no. 4, p. 295, 2013, doi: 10.1504/IJMIC.2013.053535. [6] C. P. Obite, N. P. Olewuezi, G. U. Ugwuanyim, and D. C. Bartholomew, "Multicollinearity Effect in Regression Analysis: A Feed Forward Artificial Neural Network Approach," Asian J. Probab. Stat., pp. 22–33, Jan. 2020, doi: 10.9734/ajpas/2020/v6i130151. [7] W. Wang et al., "Constructing Features for Detecting Android Malicious Applications: Issues, Taxonomy and Directions," IEEE Access, vol. 7, pp. 67602–67631, 2019, doi: 10.1109/ACCESS.2019.2918139. [8] B. Rashidi, C. Fung, and E. Bertino, "Android malicious application detection using support vector machine and active learning," in 2017 13th International Conference on Network and Service Management (CNSM), Tokyo, Nov. 2017, pp. 1–9. doi: 10.23919/CNSM.2017.8256035. [9] 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. Inform., vol. 14, no. 7, pp. 3216–3225, Jul. 2018, doi: 10.1109/TII.2017.2789219. [10] G.

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, and J. Blasco, “Dendroid: A text mining approach to analyzing and classifying code structures in Android malware families,” Expert Syst. Appl., vol. 41, no. 4, pp. 1104–1117, Mar. 2014, doi: 10.1016/j.eswa.2013.07.106. [11] M. Magdum, “Permission based Mobile Malware Detection System using Machine Learning Techniques,” vol. 14, no. 6, pp. 6170–6174, 2015. [12] M. Qiao, A. H. Sung, and Q. Liu, “Merging Permission and API Features for Android Malware Detection,” in 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), Kumamoto, Japan, Jul. 2016, pp. 566–571. doi: 10.1109/IIAI-AAI.2016.237. [13] D. O. Sahin, O. E. Kural, S. Akleylek, and E. Kilic, “New results on permission based static analysis for Android malware,” in 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Mar. 2018, pp. 1–4. doi: 10.1109/ISDFS.2018.8355377. [14] A. Mahindru and A. L. Sangal, “MLDroid—framework for Android malware detection using machine learning techniques,” Neural Comput. Appl., vol. 33, no. 10, pp. 5183–5240, May 2021, doi: 10.1007/s00521-020-05309-4. [15] X. Su, D. Zhang, W. Li, and K. Zhao, “A Deep Learning Approach to Android Malware Feature Learning and Detection,” in 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, Aug. 2016, pp. 244–251. doi: 10.1109/TrustCom.2016.0070. [16] K. A. Talha, D. I. Alper, and C. Aydin, “APK Auditor: Permission-based Android malware detection system,” Digit. Investig., vol. 13, pp. 1–14, Jun. 2015, doi: 10.1016/j.diin.2015.01.001. [17] A. Mahindru and P. Singh, “Dynamic Permissions based Android Malware Detection using Machine Learning Techniques,” in Proceedings of the 10th Innovations in Software Engineering Conference, Jaipur India, Feb. 2017, pp. 202–210. doi: 10.1145/3021460.3021485. [18] U. Pehlivan, N. Baltaci, C. Acarturk, and N. Baykal, “The analysis of feature selection methods and classification algorithms in permission based Android

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

malware detection," in 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), Orlando, FL, USA, Dec. 2014, pp. 1–8. doi: 10.1109/CICYBS.2014.7013371. [19] M. Kedziora, P. Gawin, M. Szczepanik, and I. Jozwiak, "Malware Detection Using Machine Learning Algorithms and Reverse Engineering of Android Java Code," Int. J. Netw. Secur. Its Appl., vol. 11, no. 01, pp. 01–14, Jan. 2019, doi: 10.5121/ijnsa.2019.11101. [20] X. Liu and J. Liu, "A Two-Layered Permission-Based Android Malware Detection Scheme," in 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, Oxford, United Kingdom, Apr. 2014, pp. 142–148. doi: 10.1109/MobileCloud.2014.22. [21] "Permission-Based Android Malware Detection | Semantic Scholar." https://www.semanticscholar.org/paper/Permission-Based-Android- Malware-Detection-Aung-Zaw/c8576b5df33813fe8938cbb19e35217ee21fc80b (accessed Oct. 31, 2021). [22] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," presented at the Network and Distributed System Security Symposium, San Diego, CA, 2014. doi: 10.14722/ndss.2014.23247. [23] H. Cai, N. Meng, B. Ryder, and D. Yao, "DroidCat: Effective Android Malware Detection and Categorization via App-Level Profiling," IEEE Trans. Inf. Forensics Secur., vol. 14, no. 6, pp. 1455–1470, Jun. 2019, doi: 10.1109/TIFS.2018.2879302. [24] P. Rovelli and Ý. Vigfússon, "PMDS: Permission-Based Malware Detection System," in Information Systems Security, vol. 8880, A. Prakash and R. Shyamasundar, Eds. Cham: Springer International Publishing, 2014, pp. 338–357. doi: 10.1007/978-3-319-13841-1_19. [25] M. S. Alam and S. T. Vuong, "Random Forest Classification for Detecting Android Malware," in 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing,

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

Beijing, China, Aug. 2013, pp. 663–669. doi: 10.1109/GreenCom-iThings-CPSCom.2013.122.

Applied GIS

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023

Applied GIS

ISSN: 1832-5505
Vol-11 Issue-04 oct 2023