SSHJ Everant ISSN-2456-2653
http://sshj.in/index.php/sshj
Research Article
Social Science and Humanities Journal
CrossMark

Malware Analysis on Android Apps: A Permission-Based Approach

Ariel O. Gamao

Abstract:-*The use of Android devices nowadays is almost inevitable. Having been able to get a big slice of the mobile operating systems, Android has become a wide target for malware attacks. Malware detection analysis in this study is done to contribute to the many various ways in doing the malware analysis using classification algorithm using Random Forest and Naive Bayesian. This study used a static method of analyzing and detecting malware applications through the permission requests made by each Android application as analyzed by Virus Total website. This study utilized fifty actual Android samples down-loaded from the Internet in which the samples were composed of twenty-five benign apps and twenty-five malware applications.*

Keywords:-*Classification algorithm, static method, permission requests, online validation, data collection, benign applications and malware applications*

I. INTRODUCTION

Today, users communicate each other and share things on unknown networks without knowing of the risks pertaining to their privacy, confidentiality and accessibility of information in the global world of mobile technology. Android platform is the most popular mobile operating system among the other users of Android devices, its open and dynamic environment allows a large community of developers to upload and download applications. Such extensive usage makes it an easy target for attack and misuse. An pernicious application might steal those private information of users and upload it to a specific server, which will be a risk with user's security [1].

In the Android environment, access permission is essential of an application, without which no installed apps can its worth when not granted with certain access permissions. Each android app declares its permissions during the installation phase. Android allows each app to function on operations based on its access grants as declared. However, no matter how fulfilling this may seem, it might have some flaws. Using the permissions an app is performing those operations in background which we would not have permitted it to do voluntarily. For example, a gaming app requests permissions to read contacts; and access the Internet, then there is a possibility that the said app reads the devices contact and send the data to third party servers over internet for whatever purpose. So, if the app requests for to send SMS messages, This Might permit the app to send message around as if the user sends it and eventually charge the owner on his phone billing[2].

Most of the Android malware detection approaches are concentrated on superficial features such as requested or used permissions, which can't reflect the essential differences between benign apps and malware. However, a quantitative computation model of the application risks-based on the key observation that the essential differences between benign apps and malware actually depends on the manner how those permissions were being used, or rather the way how those corresponding permission methods are used[3].

Recently, according to the IDC in 2017, phone companies shipped a total of 344.3 million smartphones worldwide in the first quarter of 2017. With the impressive market share and a great number of users worldwide, Android has become a center stage for malicious apps that attack valuable information in personal devices [4]. Malwares can cause a serious threat to Android users anywhere in the world. Studies show that more than 70 percent of smart phone apps request additional permissions

beyond their purpose. For example, a puzzle game app may request for SMS and phone call permissions that is beyond the actual purpose of the apps. During apps installation on Android, users opt whether permissions will be granted or not and this is happening in current Android architectures. However, there were many instances where this architecture has been proven ineffective to protect users since they had always been in a rush through their decisions during the installation and grant all the apps the permissions they desire [5].

Both benign and malware Android applications badly needed a set permission requests for apps to do its job on the Android devices. With this, various sets of permission as common to benign and malware apps respectively. Previous study indicated that some of permission requests for malicious applications are the following: permissions on sending SMS, receiving SMS, reading SMS messages, reading phone state, installing and deleting packages, accessing the Internet, changing phone configuration [6]

Malwares are found to have the acts of sabotage and attempts on the system based on software running behavior and its status. Based on research, malware on smartphones are relatively small compared to traditional desktop computers [7].

This study generally focused on the permission-based malware analysis on Android Apps. This specifically tried to investigate on the following points such as: what are the available Android apps (.apk) being used in the dataset? What permission sets were requested per applications as subject in this study? And what algorithm to use in the classification of benign and malicious applications?

The main objective of this study was to analyze the dataset composed of benign and malware Android applications (.apk). Specifically, the study aimed to download apps (.apk) com- posed of benign and malware; identify permission set of each application (.apk); and explore appropriate classification algorithm in the classification of benign and malicious apps based on the dataset.

II. REVIEW OF RELATED LITERATURE

This study made basis the so many studies relating to malware detection analysis on Android Apps particularly us- ing the static analysis method. The following were divided into permission-based detection, malware applications, static analysis, and classification of Apps using machine learning methods.

A. Permission-Based Detection

Many researches being conducted describing the increasing threats to Android systems that had been brought by malwares. Android architectures use a system based on permissions to control how apps behave through their access to devices and stored data

[2]. Smartphone operating systems (OSs) nowadays have been designed primarily on how security and privacy can be strengthened. One of the methods used in these systems to protect users is through the permission-based system, in which requiring the apps developers to declare what resources each application will use. If such permission request has not been granted, apps is unable to access certain resources as requested during runtime. As this permission- based systems become more common, questions have been raised about their design and implementation [8].

In order for apps to have a controlled access to different system resources, permission-based security architecture should be implemented. The boldness of the permission set plays a vital role in providing the apps the appropriate access rights in the Android environment [9]. There can be possibilities for developer to be more liberal when assigning permissions to each application in order to prevent the application to possibly face an exception. In addition, there is a possibility that the online documentation for the Android API is incomplete which would further complicate the process of assigning the strictest permission set to an application. With this, a static analysis on Android applications had been performed in order to determine the precise permission sets required by each application based on its operations [10]. On the other hand, there are third-party Android

applications with extensive APIs that include device resources, settings, and user data. Access to privacy and security-relevant parts of the API is controlled with an install-time application permission system [11].

The open source mobile platform allows the developers to develop apps that can be readily installed by Android users. Having given the users to install the third-party apps tends to have serious security issues. However, the existing security

Mechanism in Android allows a mobile phone user to see which resources and application requires [12]. Nowadays, browsers and smartphone operating systems consider apps as mutually cannot be trusted as possibly malicious in nature. Thus, applications can be secluded except for obvious inter-application communication (IPC) channels and considered odd by default, meaning apps requires user's permission for access rights as privileges. Even if applications among themselves may communication each other to support useful collaboration whereby posing a risk of permission redelegation. This happens when an apps with permissions performs a privileged task for an application having no permissions [13].

B. Malware Applications

Few of the Android applications are seldom granted with privilege to access the device resources including the sensitive ones. This scenario makes it difficult for remote access points to put any trust of network connections originating from users device. Even on the phone, different applications with diverse set of privileges can communicate with one another [14]. Android operating system has increased its popularity where a big share of the smartphone devices nowadays run on it. Recently, permission-based architecture on Android is vulnerable to apps level privilege escalation attacks. For example, Android apps may gain indirectly privileges to perform illegal operations [15]. Android apps can collaborate with the other apps based on its intent. At the same time it can also control personal information or use permissions granted by a user while users are unable to detect when certain apps communicate with other apps. Thus, users might not be aware of any information leakage if an app happens to be malware [16].

Security on Android devices has been built with the use of permission requests for third-party apps to gain access to certain Android resources. When the user accepts the permissions sets of request, the installation proceeds with the installation. This operation allows the user to be aware of the risks of the resources to be accessed by the third-party apps. Although the users are aware of the permission sets, still the user's don't know what the apps bring in terms of the dangers or threats to the device. [17]. Android operating system being the most popular platform nowadays has become the main target for various attacks due to its increasing number of users. In analyzing malwares, a static analysis approach has been done on permission sets requested by third-party apps. [18].

Most of the Android apps can be downloaded from the Android markets for free. But there is no guarantee that those apps are free from malicious intentions. In detecting malware apps, a machine learning-based malware detection system can be designed to enhance privacy and security of Android users. The system uses permission-based architecture to analyze the said features with the use of machine learning classifiers to classify whether the application is benign or malware [19].

Studies have revealed that current families of Android malware are difficult to promptly spot in the wild. This is because of the evasion techniques being used to conceal malicious payload, usually within seemingly innocuous apps that provide functionalities that users want. By employing polymorphic techniques and encrypting malicious payload, signature-based scanning is easily bypassed. With increased code obfuscation, malware analysts take longer to uncover the malicious behavior, classify samples, and generate signatures for detecting the new threats. Moreover, some Android malware families like An server Bot are known to have the capability to fetch and execute malicious payloads at run time thus rendering the zero-day detection of such malware by prior signatures quite ineffective [20].

C. Static Analysis

In analyzing malware apps using static analysis, efforts are concentrated on requested permissions. Permission combinations can be suspicious or dangerous when apps request permissions other than what is expected from it by increasing the number of permissions to define more permission combinations. However, if there are limited variances in requested or granted permissions between benign (good) apps and malware, permission- based methods may experience the problems of low rate of detection [21].

Static analysis refers to features that are collected without executing the code. In Android malware can rely on Java byte code extracted by disassembling an application. The manifest file is also a source of information for static analysis. One disadvantage of static analysis is that it is blind to dynamic code loading, that is, static analysis fails to deal with parts of the code that are downloaded during the execution. In contrast, dynamic analysis can examine all code that is actually executed by an application [22].

D. Classification of Apps using Machine Learning Algorithms

During the installation of the third-party apps, the running processes of android apps were analyzed to establish the suitable methods for feature selection and algorithms for classification to be used for the analysis and detecting malwares. Four methods for feature selection using attribute- based and subsets-based selection methods that were reduced the attribute numbers and improve the performance of algorithms for classification. The performances of the selected algorithms for classification such as Bayesian, decision tree, and SVM were the compared. Furthermore, the size of the dataset was analyzed to quantify the progress of the classification algorithms. The permission sets of the malicious applications were also quantified in order to use clustering analysis [23].

The obtained results from the analysis of permissions showed that among the family of classification algorithms, Bayesian algorithm yielded an unfavorable result. Generally, the best result among the chosen classification algorithms was Random Forest algorithm where a population of 100 trees, 87 percent was achieved in terms of malicious apps detection accuracy with 0.95 for area under the curve (AUC) [24]. Machine learning (ML) classifiers have played a part in the development of intelligent systems for several domains over the years. ML approaches are gaining traction in identification and detection of malware on both mobile and PC platforms. Our work is based on supervised machine learning whereby the features described in the previous section are acquired from a labelled dataset and used to build and train a model. The ML algorithms considered in our investigation include: Decision Tree (tree-based), Simple Logistic (function-based), Nave Bayes (probabilistic), PART (rule-based), and RIDOR (rule-based) [25].

To detect malware applications on Android, a study on a machine learning approach relies on K-Nearest Neighbor classifier to train the model with features such as incoming/outgoing SMS and calls, Device status and running applications/processes. In addition, a framework which relies on machine learning algorithms for Android malware detection using features obtained from Android events and permission based to learn and classify malware and benign applications [26].

In a study, tools can be built that can warn users about applications that request permissions that were blacklisted. Black- listed patterns were manually requested to represent malicious sets of permissions. In contrast, a statistical whitelisting approach has been implemented to warn users about applications that do not match the permission request patterns expressed by high-reputation applications. These two approaches were complementary; human review of the statistically-generated patterns could potentially improve them [27].

III. METHODOLOGY

This section covers the methods and techniques used in the conduct of this study in the analysis of Android Apps.

1) **Data Gathering:** As show in figure 1 that the researcher collected the sample Android apps by downloading twenty- five benign Android Apps and twenty-five are malware Apps. In order to validate the accuracy of classified apps, the said apps were one-by-one manually submitted to the VirusTotal website for online classification to make sure the classification of both apps. Every after submission of each Android application (.apk), the researcher individually

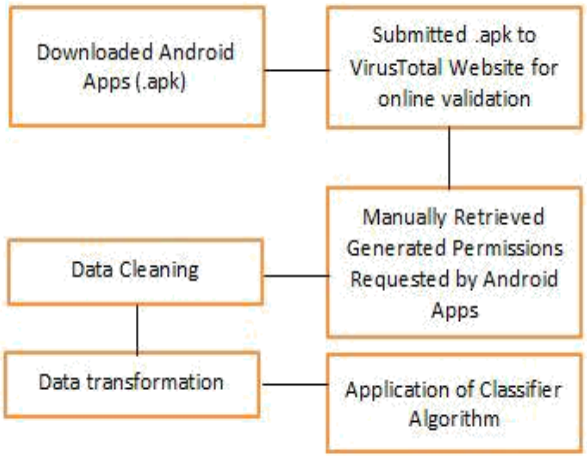

Fig.1. Operational Framework of the Study

Retrieved the generated requested permissions used in each app (apk). After the creation of a dataset, the researcher performed data cleaning to make sure only the needed data were included. Data were transformed into a format the researcher desired. In order to cross-validate the classification process, the dataset was divided into training set and testing set.

Figure 2 demonstrates the online validation of benign applications through Virus Total. Here, the researcher submitted all the benign applications to Virus Total website for online validation of each benign application through manually submission. First, the apps (.apk) were submitted individually, then clicking the process. After which, the said website started to validate and scan each Android application using various legitimate malware and spyware scanners developed by various renowned companies worldwide such as Avast, AVG, ESET-NOD32 and many more. When the apps are free from malwares, each scanning engine will certify the said apps is clean. Therefore, the apps is not a malware but benign.

When the malware applications were individually submitted to the Virus Total website, all the available malware and spyware scanning engines detected several malicious codes embedded on the said applications. When the system detected such malwares, on the detection tab, each engine provided a red-colored notification. This means a warning to the user that the upload APK file is a malware. Therefore, the downloaded malware applications were validated for accuracy testing which were subject in the testing later on.

2) **Dataset Splitting:** The dataset of the entire fifty (50) apps (.apk) were divided into training and testing sets in a 80/20 percentage ratio for training and testing sets. As shown in figure 4 that in the training data, benign apps got a frequency of nineteen (19) while three (3) for the malware. However, in the testing set, malware got a higher frequency by twenty-two (22) compared to benign which only got a frequency of nine (6). The dataset containing the attributes of permissions, those that were requesting permissions were assigned with ”YES”, and ”NO” for those attributes that were not requested.

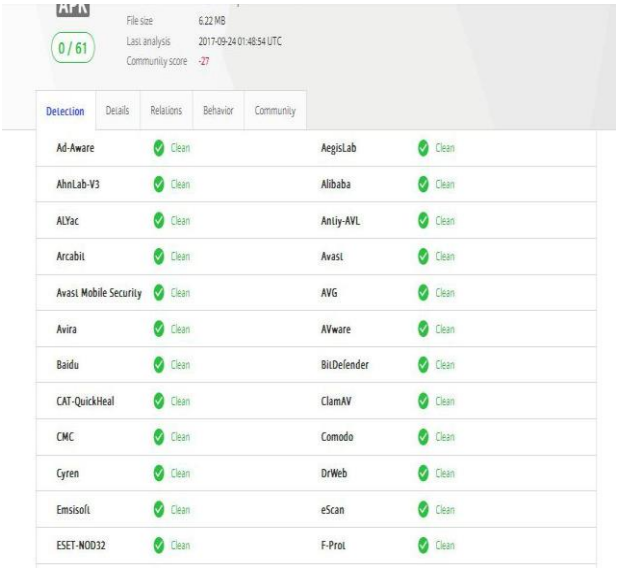

Fig.2. Online validation through Virus Total for Benign Apps

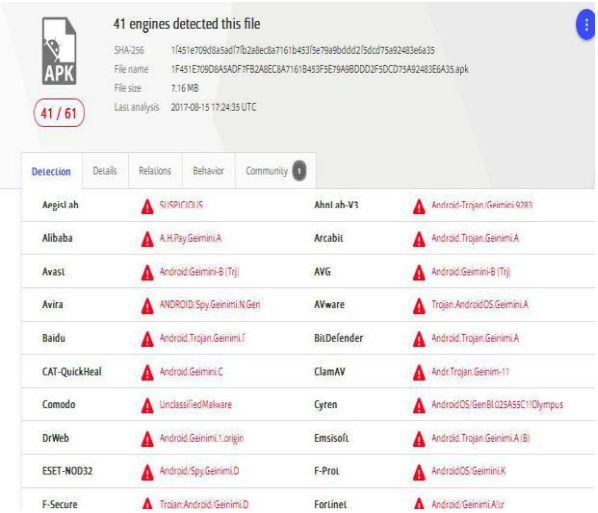

Fig.3. Online validation through Virus Total for Malware Apps

IV. RESULTS AND DISCUSSION

The online malware classification validation through the Virus Total was able to generate other details of each Android application (.apk) like countries of origin. Figure 5 shows that sample applications originated from the seven (7) countries where most of them originated from the USA for both the benign and malware. The apps second from the top originated in China along with from unknown countries.

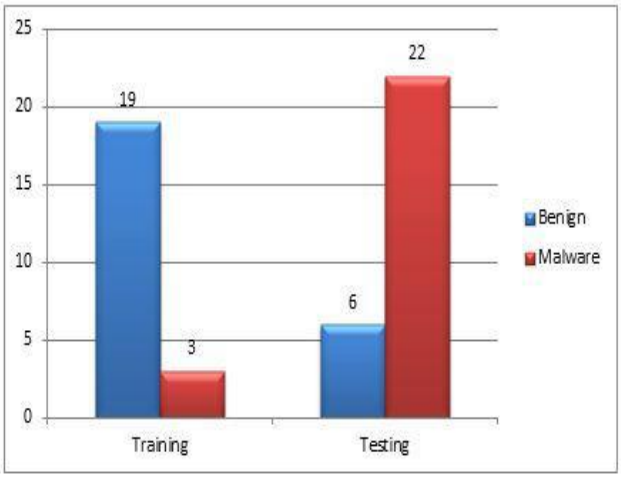

Fig.4. Training and Testing Sets

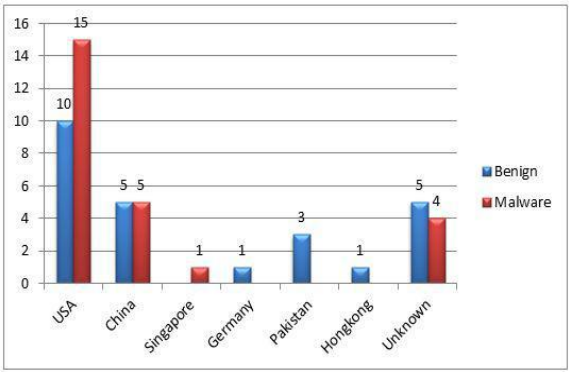

Fig.5. Classification of Android Apps by Country of Origin

Figure 6 shows the top 15 permission requests of the benign applications where all of the benign Android applications requested permission to access the Internet whereby allowing application to create network sockets and this followed by requested permissions to view the status of all networks. The last three requested permissions were allow the applications to read all phone contacts, managing phone accounts, and a permission in allowing the applications to access location resources especially the global positioning system (GPS).

The results shown in figure 7 are another set of top fifteen (15) permission requests made by malicious applications. Not all of the malwares accessed the Internet, unlike the benign apps. The permission second from the top is allowing the applications to send SMS messages. According to the Virus Total that when applications are permitted to send SMS messages, the said applications may send messages to target receiver even without the owner's knowledge and this may add charges to the owner's bill [28].

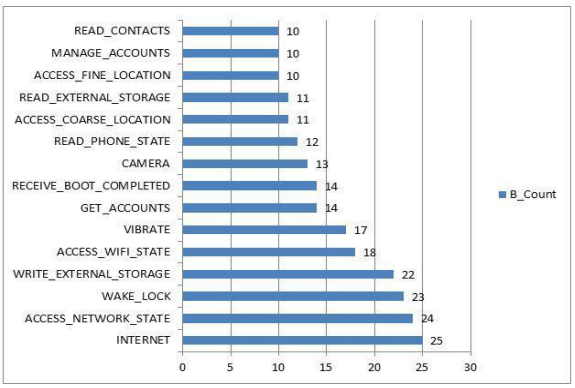

Fig.6. Top 15 Benign Android Apps

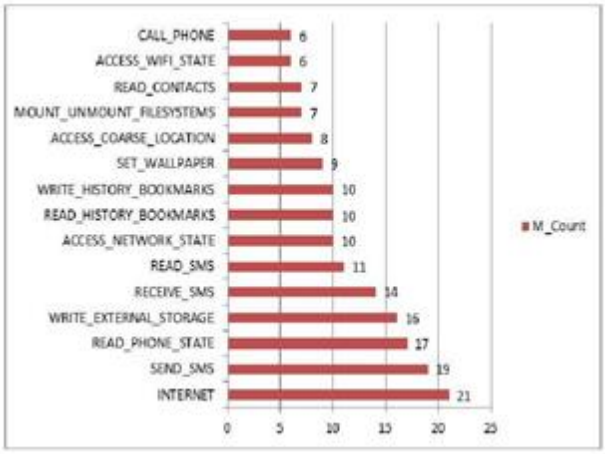

Fig.7. Top 15 Malware Android Apps

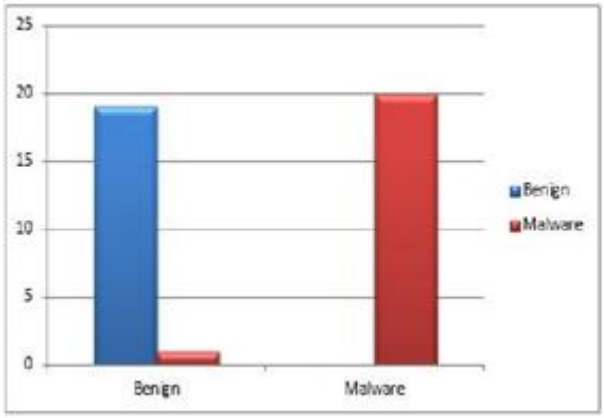

Fig.8. Common Permissions for both Benign and Malware

When both the top fifteen permission requests for both benign and malware applications were put together, as shown in figure 8 the researcher was able to come-up with twenty-three (23) combined permissions themes. Most of the permission requests made by the malware when compared to the benign applications were actually the set of permissions as identified by a previous study [29]

When the random forest method was executed using as R simulation as shown in figure 9 for the training dataset, it was observed that there was one (1) malware apps that was misclassified as the benign. This means that there was a 4.8 percent error using the Random Forest classifier as identified to be one apps. Based on the original dataset, there were twenty-one malware and nineteen benign applications. The total numbers of attributes in the dataset were eleven (11). Good enough for the classifier algorithm to perform its operations.

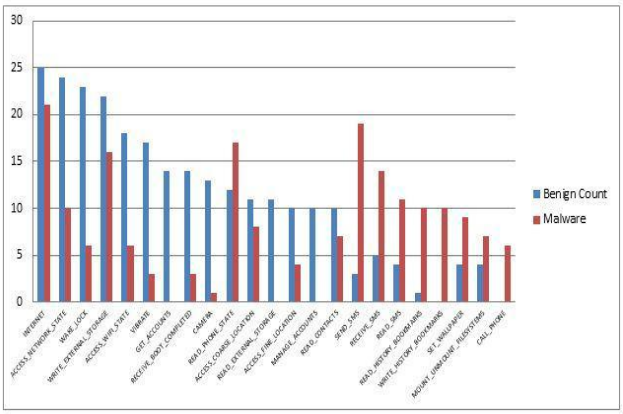

Fig.9. Confusion Matrix for Benign and Malware Apps using Random Forest Classifier Algorithm

Figure 10 shows the confusion matrix of results on the simulation using Naive Bayesian where two (2) applications were misclassified as benign which resulted to a 9.5 percent of error in the classification of malware apps. When this happens the classification algorithm incurred and error via misclassification.

For the test of accuracy for both classifier algorithms namely: Random Forest and Naive Bayesian, the graph on figure 11 shows that Random Forest Method is more accurate than the Naive Bayesian algorithm. When the simulation was done on R language, Random Forest got 97.5 compared to 95 percent for the Naive Bayesian. The Researcher tried to adjust the training and testing sets ratio, and it turned out that the results still the same.

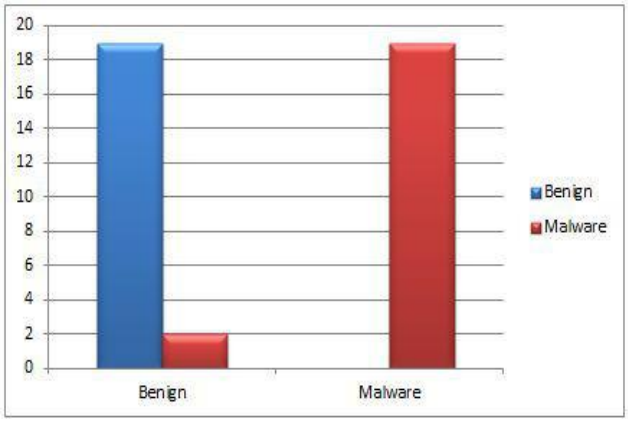

Fig.10. Confusion Matrix for Benign and Malware Apps using Naive Bayesian Classifier Algorithm

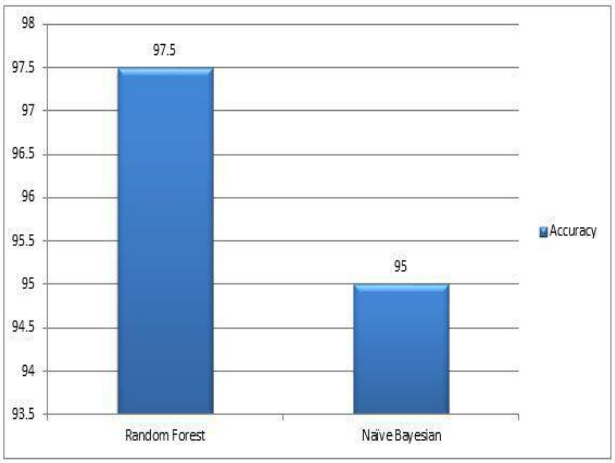

Fig.11. Test of accuracy using Random Forest and Naive Bayesian Algorithms

V. CONCLUSION AND RECOMMENDATION

This study shows that malware detection analysis can be applied with random forest and Nave Bayesian machine learning classification algorithms. The malware scanning can be done through online via Virus Total in which this operation is for free without a need of developing tools to be used for the extraction of the permissions requested by each of the Android apps.

Having been able to implement the classification method using Random Forest and Nave Bayesian on a malware detection analysis, also other effective classification algorithms can also be used for the same purpose for further reliable tests.

In order to improve the malware detection analysis, it is suggested to have more data in the dataset still subject for verification through a very reliable malware classification tools available on the web.

ACKNOWLEDGMENT

The researcher has gratefully acknowledged Virus Total website in providing a very powerful tool in extracting features in the form of permission sets from the downloaded Android applications used for this research work.

REFERENCES

1) Koundel, D., Ithape, S., Khobaragade, V., Jain, R. (2014). Malware Classification using Nave Bayes Classifier for Android OS, 5963.

2) Wei X, Gomez L, Neamtiu I. and Faloutsos M., ”Permission evolution in the Android ecosystem,” in Proc. of Computer Security Applications Conference, 31-40, 2012. Article (CrossRefLink).

3) Ye, Y., Wu, L., Hong, Z., Huang, K. (2017). A risk classification based approach for Android malware detection. KSII Transactions on Internet and Information Systems.

4) IDC: Smartphone OS Market share http://www.idc.com/promo/smartphone-market-share/os. July 29, 2017. 2017 Retrieved:

5) Rashidi, B., Fung, C. (2016). XDroid: An Android Permission Control Using Hidden Markov Chain and Online Learning.

6) Verma, S., Muttoo, S. K. (2016). An Android Malware Detection Framework-based on Permissions and Intents, 66(6), 618623.

7) Zhao, M., Zhang, T., Wang, J., Yuan, Z. (2013). A smartphone malware detection framework based on artificial immunology. Journal of Networks.

8) Au K W Y, Zhou Y F, Huang Z, Lie D., ”PScout: analyzing the Android permission specification,” in Proc. of the 2012 ACM conference on Computer and communications security. ACM, 217-228, 2012. Article (CrossRefLink).

9) Johnson R, Wang Z, Gagnon C and Stavrou, ”A. Analysis of Android Applications’ Permissions,” in Proc. of IEEE Sixth International Con- ference on Software Security and Reliability Companion. 45-46, 2012. Article (CrossRefLink).

10) Barrera D, Kayacik, H. G, Van Oorschot P C and Somayaji A., ”A methodology for empirical analysis of permission-based security models and its application to android,” in Proc. of ACM Conference on Computer and Communications Security, CCS

2010, Chicago, Illinois, USA, October. 73-84, 2010. Article (CrossRefLink).

11) Felt A P, Chin E, Hanna S, Song D and Wagner D., "Android permissions demystified," in Proc. of ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October. 627-638, 2011. Article (CrossRefLink).

12) Nauman M, Khan S, Zhang X., "Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints," in Proc. of ACM Symposium on Information, Computer and Communi- cations Security, ASIACCS 2010, Beijing, China, April. 328-332, 2010. Article (CrossRefLink).

13) Felt A P, Wang H J, Moshchuk A, Hanna S and Chin E., "Permission re-delegation: attacks and defenses," Usenix Conference on Security. USENIX Association, 22-22, 2011. Article (CrossRefLink).

14) Dietz M, Shekhar S, Pisetsky Y, Shu A and Wallach DS.,"Quire: lightweight provenance for smart phone operating systems," Dissertations Theses, 23-23, 2011. Article(CrossRefLink)

15) Bugiel S, Davi L, Dmitrienko A, Fischer T and Sadeghi AR.,"XManAndroid: A new Android evolution to mitigate privilege escalation attacks," Technical Report, Technische Universitat Darmstadt, TR-2011- 04, 2011. Article (CrossRefLink).

16) Sakamoto S, Okuda K, Nakatsuka R and Yamauchi T.,"DroidTrack: tracking and visualizing information diffusion for preventing information leakage on Android," Journal of Internet.

17) Tchakounte, F. (2014). Permission-based malware detection mecha- nisms on Android: analysis and perspectives. Journal of Computer Science and Software Application, 1(2), 6377.

18) Seth, R., Kaushal, R. (2015). Permission based Malware Analysis Detection in Android.

19) Sakamoto, S., Okuda, K., Nakatsuka, R., Yamauchi, T. (2013). DroidTrack: Tracking information diffusion and preventing information leakage on android.

20) A.pvrille and T. Strazzere, Reducing the window of opportunity for Android malware Gotta catch em all, Journal in Computer Virology vol. 8, No. 1-2, pp. 61-71, 2012.

21) Ye, Y., Wu, L., Hong, Z., Huang, K. (2017). A risk classification based approach for Android malware detection. KSII Transactions on Internet and Information Systems.

22) Kapratwar, A. (2016). Static and Dynamic Analysis for Android Mal- ware Detection.

23) Uur, P. (2014). Permission-Based Malware Detection Analysis in An- droid Applications.

24) Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Nieves, J., Bringas, P. G., lvarez Maran, G. (2013). Mama: Manifest Analysis for Malware Detection in Android. Cybernetics and Systems, 44(67), 469488. https://doi.org/10.1080/01969722.2013.803889.

25) Yerima, S. Y., Sezer, S., Muttik, I. (2014). Android Malware Detection Using Parallel Machine Learning Classifiers, (Ngmast), 1014.

26) Braehler S (2010). Analysis of Android Architecture.

27) Frank, M., Dong, B., Felt, A. P., Song, D. (2012). Mining Permission Request Patterns from Android and Facebook Applications.

28) Online Malware/Spareware Scanning https://www.virustotal.com. Re- trieved: Sept 1, 2017.

29) Sato, R., Chiba, D., Goto, S. (2013). Detecting Android Malware by Analyzing Manifest Files. Proceedings of the Asia-Pacific Advanced Network, 36, 23. https://doi.org/10.7125/APAN.36.4.

30) https://www.educba.com/text-mining/. Date Retrieved: August 25, 2016.

Ariel O. Gamao is a DIT student of Technological

Institute of the Philippines,Quezon City. The author is currently enrolled of a DIT Program through an off Site campus hosted By the University of Mindanao,Matina, Davao City.

Mr. Gamao is also a faculty of the Davao del Norte State College, Panabo City Who at the same time a Ph.D in Educational Graduate.