International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-4, November 2019

An Effective Performance Based on Static and Dynamic Features to Detect Malware in Android by Machine Learning Algorithm

P.S.Rajakumar, V. R. Niveditha, T. V. Ananthan, N. Kanya

Abstract: ***Android is dominating the smartphone market with more users than any other mobile operating system. Yet concern from hackers has also risen with its growing popularity, as the number of malicious applications remains to rise. Wide-ranging work on malware identification and prevention for Android devices has been carried out in recent years, although Android has also introduced numerous security measures to fix malware complications, containing Unique User ID (UID) for every request, software permissions, and its Google Play scattering platform. In our proposed work, incorporates the analysis of static and dynamic features of these applications with the goal of evaluating their actions by exploring different attributes such as authorization, use of CPUs and storage utilization. In this paper machine learning methods to evaluate the comparative efficiency of extracted static and dynamic features to identify AM. This results is quite powerful and demonstrates the AUC of 0.972 can be used to identify AM with a high degree of accuracy than the dynamic function.***

Keywords: static features, dynamic features, Machine Learning (ML) algorithm, Android Malware (AM)

I. INTRODUCTION

Android is the most popular mobile computing device OS like tablets and smartphones. Mobile devices contain sensitive user information, which is why malware is being established to steal such information. A recent study shows that the most common mobile or tablet OS has Android retains 70% to 80% of the market share. Since its first release in 2008; about 50 billion applications have been downloaded and are launched every month about 20,000 applications. There has been an unprecedented rise of AM in recent years due to the rise in the many distrusted outlets and pages which offered profitable yet harmful apps. Such malware programs can be for any kind, like Ransomware, or they can build a loophole for stealing personal information. Even has a lot of faults the Google Play Store is not free. Google is trying to get rid of malicious apps with Google Bouncer, but it allows for uploading false or corrupted devices is the some drawbacks. Apart from Google Play, there are many third-party software stores have little influence and provide Android apps over the publication of apps namely App China and others. It is very important to learn about their behavior to identify malicious applications. This can be done on these applications by performing analysis of malware. There are two step in Malware detection, first one is used to obtain as much information as possible before running the program, and a static analysis is conducted. Different data properties are evaluated and analyzed. Singh et.al (2018) dynamic analysis is performed in the second step which has been controlled and simulated environment simultaneously to detect and capture malware's runtime behavior by running different tools. Malicious programs have been conducted in a Geny motion environment in our experimental tests and various tools namely Wireshark, OS Manager have been used for real-time surveillance. In this paper to define its behavioral pattern through a static and dynamic approach to analyze AM.

Manuscript published on November 30, 2019.
* Correspondence Author
P.S.Rajakumar *, Professor Department of Computer Science and Engineering. Dr. M.G.R. Educational and Research Institute, Chennai 600095, Tamil Nadu, India. Email:suraarus@yahoo.com.
V. R. Niveditha, Research Scholar Department of Computer Science and Engineering. Dr. M.G.R. Educational and Research Institute,Chennai 600095, Tamil Nadu, India.
T.V.Ananthan, Professor, Department of Computer Science an dEngineering. Dr. M.G.R. Educational and Research Institute,Chennai 600095, Tamil Nadu, India.
N.Kanya, Professor Department of Computer Science and Engineering. Dr. M.G.R. Educational and Research Institute,Chennai 600095, Tamil Nadu, India.

II. LITERATURE REVIEW

There have been sufficient studies in this area of malware detection for Android in recent years. Such research can be classified loosely into two groupings namely dynamic analysis and static examination. Burguera et.al (2011) collects as a feature set and tracks the program process calls and then exploits malware identification and detection based on clustering algorithms. Wu et.al (2014) extracts by n-gram the structures of the sequences of the API calls composed from applications to complete the classification during runtime. Saracino et.al (2018), framework of host-based malware detection, which constructs the detection system by tracking features at various Android stages like client, device and kernel. Tamet.al (2017) the requirement of dynamic evaluation is to execute a system and interpret findings that could counter ambiguity and dynamically packed data, but deliver limited coverage of software and spend additional time and processing assets. Static analysis delivers quantitative results which are more effective and precise compared to dynamic analysis, but it is difficult to manage with ambiguity and loaded dynamic technology. In order to complete the identification function, Aafer et.al(2013) focuses on the top-169 API calls in malware which are more common than in harmless collections.

Wu et.al(2012) described classifies the effects of the clustering via KNN by using the clustering algorithm to handle static attributes namely API requests, permissions and attempts. Arp et.al(2014) extracts various features contains network addresses, API calls, critical device permissions for using SVM to conduct classification. Sun et.al(2018), based on the analysis of permission with an efficient detection system use to manage with the quick growth in the amount of malware on Android. Kumar et.al(2019) suggested a new way of distinctive between the applications of malware and benign based on the permission mining association rule.Houet.al(2017) suggests a new technique of feature extraction to classify the linguistic communication of applications that uses meta path and API calls to identify malware by incorporate the SVM algorithm. Therefore, for static analysis, Opcode is also a widely used resource for analysis. The approaches has both use Opcode n-grams as a function abstraction suggested in paper shabtai et.al(2010) and Canfora et.al(2015) to define software functionality. Zhang et al.(2018), suggest DalvikOpcode's weighted likelihood graph and features extract topology as software demonstration, and then look for correlations with these system features to detect AM.

III. AM DETECTION

Applications for AM contain mainly of Trojans when user interfaces or using of icons which resemble a harmless program, a standard Android Trojan can trick the consumer. Throughout the setup, Android Trojans also show a service level contract which finds the privileges from a client to access personal information, like phone number. For example, the Trojan can submit SMSs in the background to the premium rate numbers. Also often used as spyware is Android Trojans. These malicious programs can reach the private information of a client and also send it to an isolated server. Such spyware's main resolve is to steal the information namely passwords, phone locality, contacts, credit card or bank particulars, online browsing, text messages, etc. Botnet competencies may also be involved in additional sophisticated implementation.

This system can classify as static or adaptive depending on the features used to describe an application. By operating an application, static analysis is done. Definitions for static features contain (a) privileges (permission), (b) API calls that can be derived from the directory of Android Manifest.xml. Dynamic evaluation contracts with features derived from the software when operating, containing(a) memory use (b) CPU usage. The remainder of this section describes the features extracted from the algorithm used for application and ML.

A. *Static analysis*

In the static analysis, the programs are evaluated and the functions are retrieved without the program being run on a simulator or computer. Compared to other analytical techniques for the detection of AM, static analysis devours less resources and time because it does not include the application being executed. The main drawback of this research is software complication, which creates its challenging to detect the applications of malicious behavior as it is not conceivable to match designs. Baskaran and Ralescu(2016)this review could classify errors in runtime, logical variations, and potential breaches of protection. The authorization and API calls are the most widely used static functions.

Android software is a zip packet in the shape of an Android download, or apk, file. Together with numerous other tools and directories, the apk repository contains the manifest. To remove the interesting features, we first essential to converse the apk files which has to be using the Virustotal APK method. The AndroidManifest.xml file comprises a number of features that might be used for static analysis. Here focusing on the application's request for permissions. The AndroidManifest.xml lists all the permissions based on the application requires. Android uses an exclusive xml format, so in this research work have developed our own xml parser to remove authorization features from AndroidManifest.xml documents.

There is a max of 135 privileges from Google. Then constructing a binary vector from the permissions removed. This function vector is referred to as R = (r1, r2,., r135), expressed in equation(1).

$$ri = \begin{cases} 1 & if the ith permission is present \\ 0 & otherwise \end{cases} \quad \text{.....(1)}$$

The following steps that define the method to remove the permissions features given in an Android software.

1. The Android application is the reverse engineer. Using the Virus total APK tool, this reverse engineering is attained.
2. Use our custom xml parser to extract the necessary permissions from the AndroidManifest.xml file.
3. Generate a vector of the binary function as in (1).
4. Finally, we created an ARFF file format permission vector dataset for all applications stored in our dataset.

Many of the 135 available privileges in our repositories have never been required in any of the Android apps. Such approvals have been excluded from attention as it has not apply to the study.

B. Dynamic Analysis

Dynamic analysis is a program's checking and assessment by real-time execution of results. The goal is to find errors in a system while working, instead of testing the software offline repeatedly. It is a detection method that purposes to estimate malware by running the application and the highest benefit of this technique is that it defines the behavior of the application during runtime and loads the target data. In this method of analysis, resource consumption is more like static analysis. Rao and Hande(2017)the approach of dynamic behavioral detection constructs operating system utilizing a virtual machine, sandbox and additional types pretends program accomplishment to learn the behavior model of the application.

To learn the actions of a program when it is implemented, a complex evaluation is done. Continuous monitoring is important to collect CPU and storage consumption data. Results were obtained in our paper by initially running an emulator support application (with a 64-bit processor and an unrestricted guest CPU).Therefore, along with continuous monitoring, all forms of operations that could be carried out in a project are performed. Results in the form of maximum and minimum or average values are obtained.

A similar method has been used with different tools for CPU and storage use and its definition is as follows:

C.CPU Usage

The program was mounted on an emulator in the initial step preceded by its execution. On the other side, the following order was run simultaneously on a command prompt:

adb shell top –m 10

The above command displayed the top 10 applications using the uppermost use of the CPU. The use of CPU has command continuously monitors in an emulator for numerous applications. Once the program runs on the simulator, this command continues to show the sum of CPU used by an application. The sum of CPU usage can differ with the application's various activities.Figure.1 demonstrates the use of a deceptive application called "All in One Toolbox" by the system title "immobile.toolbox.full" for the CPU.

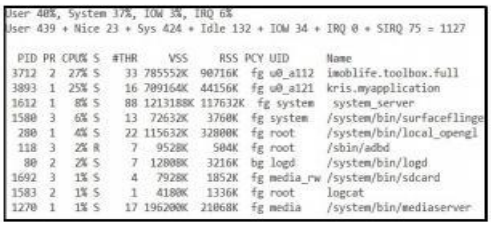

Fig.1. CPU observing toolbox of malicious application

When an application is performed, all of its operations will be conducted. For eg, all operations such as testing, RAM clearing and its other tasks are conducted in the "All in one toolbox" software and the maximum amount of CPU use was tracked at the same time. Once all the activities of the app were carried out, the maximum use of the CPU was recorded during runtime. This number is also measured the total utilization cost of the CPU and the minimum value is typically null because it requires 0 percent of the CPU if a program is not operating. We found through further testing that some harmful programs were still operating and continuing to use the processor in the context, well after which had been shut down.In Figure.1, after shutting it down many times, the "AC business" program shown as "kris.my software" continued to run in the context. CPU use was found in runtime in our report, and even if the malicious device continued to run in the background, its cost was reported in terms of maximum and minimum usage.

D.Memory Analysis

Examination of storage was carried out using an OS control software to learn the transient resources used by malicious and legitimate programs. The OS screen presented a selection of volatile memory used during runtime by applications. Steps for monitoring the use of RAM are close to tracking the use of CPU. The program was constantly accomplished and tracked on the screen of the OS. Figure.2 displays the RAM that the Incognito malicious program uses.

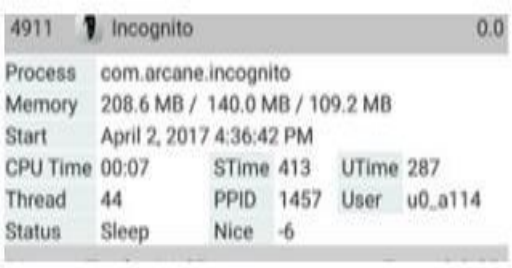

Fig.2. RAM consumption of malware Incognito

IV. RESULTS AND DISCUSSION

The experiment results have been conducted by various sets. Initially in the sense of AM detection, they conducted experiments to test the effectiveness of different ML algorithms. In this paper ML methods to calculate the comparative efficiency of extracted static and dynamic features to identify AM. Definitions for static features contain (a) privileges (permission), (b) API calls that can be derived from the directory of Android Manifest.xml. Dynamic evaluation contracts with features derived from the software when operating, containing (a) memory use (b) CPU usage. In order to evaluate the region under the ROC curve (AUC) to determine the performance of our experiments. Table.1 demonstrates the AUC values of various Weka algorithms based on (dynamic) use of memory analysis and CPUs and (static) permissions and API calls. In Figure.3, this same information is given as a bar graph.

Table.1 Comparison of ML Algorithms

ML Algorithm	Static AUC	Dynamic AUC
Naïve Bayes	0.966	0.496
Simple Logistic	0.9328	0.533
Random Forest	0.956	0.877
Sequential Minimal Optimization	0.8547	0.599

The given scatter plot of scores for cases of benign and malware, the graph of ROC curveas the threshold varies by the scoresis the TPR versus the FPR range.An AUC of 1.0 suggests the optimal situation where there is a limit dividing harmless or malicious scores entirely, whereas an AUC of 0.5 implies the binary classifier is no better than reversing a coin. In addition, the AUC can be definite as the probability that a positive occurrence selected at random achieves higher than a negative instance selected at random.

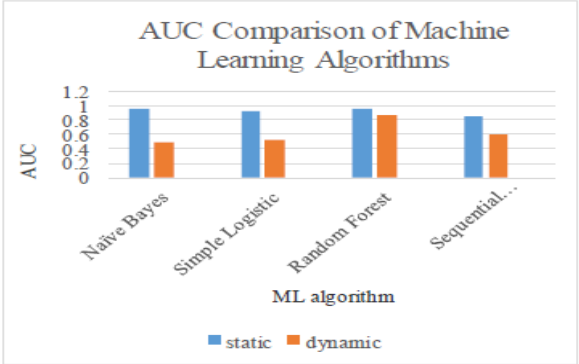

Figure.3 AUC Comparison of ML Algorithms

In this work, practice on the dynamically derived function vector which includes CPU tracking for harmful program and storage use to evaluate device calls. The extraction method for the functionality is mentioned in Section 3.2 above. To achieve an AUC of 0.884 for this test, which implies that the application and storage processing capability of the CPU alone does not produce particularly strong detection performance we also similarly assessed our (static) authorization and the function of API calls. Note that the extraction method for this functionality is defined in Section 3.1. The static feature achieves an AUC of 0.972 in this scenario. This finding is quite good and indicates that it is feasible to use a relatively simple static feature to identify high accuracy AM.

V. CONCLUSION

This paper addresses mobile malware evaluation. All needs to identify the application's malicious actions. Static research is the first phase in analysis. It is necessary for each application to go through static analysis because most malware developers change the code as well as permissions in the APK file, but analyst nowadays wants to know the application's behavior which type of event attacker wants to execute, so they need to run the application on the sandbox and examine the application's behavior. In contrast, malicious programs used less CPU but used more RAM than usual applications in dynamic analysis. The static features achieve an AUC of 0.972 for AM detection that a basic permission-based static interface is slightly more insightful than a system-based dynamic function.

REFERENCE

1. A.Saracino, D. Sgandurra, G. Dini, and F. Martinelli, ‘‘MADAM: Effective and efficient behavior-based AM detection and prevention,’’ IEEE Trans. Depend. Sec. Comput., vol. 15, no. 1, pp. 83–97, Jan./Feb. 2018.
2. P.Sakthi Shunmuga Sundaram , N.Hari Basker , L.Natrayan. Smart Clothes with Bio-sensors for ECG Monitoring, International Journal of Innovative Technology and Exploring Engineering, Volume 8, Issue 4, 2019, Pages 298-301.
3. L.Natrayan, V Sivaprakash, MS Santhosh, Mechanical, Microstructure and wear behavior of the material AA6061 reinforced SiC with different leaf ashes using advanced stir casting method, International Journal of Engineering and Advanced Technology, 2018,8(2S);366-371.
4. S.Yogeshwaran, R.Prabhu, Natrayan.L, Mechanical Properties Of Leaf Ashes Reinforced Aluminum Alloy Metal Matrix Composites, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 13, 2015.
5. D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu, ‘‘DroidMat: AM detection through manifest and API calls tracing,’’ in Proc. 7th Asia Joint Conf. Inf. Secur., Aug. 2012, pp. 62–69.
6. Natrayan L, Senthil kumar MS, “Optimization of tribological behaviour on squeeze cast Al6061/Al2O3/SiC/Gr HMMCs based on taguchi method and artificial neural network,” Journal of Advanced Research in Dynamical and Control Systems,11(7), pp. 493-500.
7. Kumar, M. Senthil, et al. "Processing and characterization of AA2024/Al_2O_3/SiC reinforces hybrid composites using squeeze casting technique." Iran. J. Mater. Sci. Eng 16.2 (2019): 55-67.
8. I.Burguera, U. Zurutuza, and S. Nadjm-Tehrani, ‘‘Crowdroid: Behaviorbased malware detection system for Android,’’ in Proc. 1st ACM Workshop Secur. Privacy Smartphones Mobile Devices, 2011, pp. 15–26.
9. Natrayan, L., M. Senthil Kumar, and Mukesh Chaudhari. "Optimization of Squeeze Casting Process Parameters to Investigate the Mechanical Properties of AA6061/Al_2O_3/SiC Hybrid Metal Matrix Composites by Taguchi and Anova Approach." Advanced Engineering Optimization Through Intelligent Techniques. Springer, Singapore, 2020. 393-406.
10. J. Zhang, Z. Qin, K. Zhang, H. Yin, and J. Zou, ‘‘Dalvik opcode graph based AM variants detection using global topology features,’’ IEEE Access, vol. 6, pp. 51964–51974, 2018.
11. K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro, ‘‘The evolution of AM and Android analysis techniques,’’ ACM Comput. Surv., vol. 49, no. 4, p. 76, 2017.
12. R. Kumar, X. Zhang, R. U. Khan, and A. Sharif, ‘‘Research on data mining of permission-induced risk for Android IoT devices,’’ Appl. Sci., vol. 9, no. 2, p. 277, Jan. 2019.
13. G. Canfora, A. De Lorenzo, E. Medvet, F. Mercaldo, and C. A. Visaggio, ‘‘Effectiveness of opcode ngrams for detection of multi family AM,’’ in Proc. 10th Int. Conf. Availability, Rel. Secur., Aug. 2015, pp. 333–340.
14. L.Natrayan, P. Sakthi shunmuga sundaram, J. Elumalai. Analyzing the Uterine physiological With MMG Signals using SVM, International journal of Pharmaceutical research, 2019, 11(2); 165-170.
15. V. Rao and K. Hande, “A comparative study of static, dynamic and hybrid analysis techniques for AM detection,” Int. J. Eng. Dev. Res., vol. 5, no. 2, pp. 1433–1436, 2017.
16. W.-C. Wu and S.-H. Hung, ‘‘DroidDolphin: A dynamic AM detection framework using big data and machine learning,’’ in Proc. Conf. Res. Adapt. Convergent Syst., 2014, pp. 247–252.
17. Y. Aafer, W. Du, and H. Yin, ‘‘DroidAPIminer: Mining API-level features for robust malware detection in Android,’’ in Proc. Int. Conf. Secur. Privacy Commun. Syst. Cham, Switzerland: Springer, 2013, pp. 86–103.

AUTHORS PROFILE

P.S.Rajakumar Completed M.C.A from Bharathiar University , Coimbatore , TamilNadu, in 2001, M.Tech in Computer Science and Engineering from Dr.M.G.R.Educational and Research Institute University in 2006, and received his Ph.D. degree in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad, Telangana, India in 2018. Published 12 papers in National and International journals. Having Membership in professional bodies like CSI, ISTE, IEEE, MIEI. Currently working as a Professor in the department of Computer Science and Engineering at Dr.M.G.R.Educational and Research Institute, Chennai, Tamil Nadu, India

Ms. V. R. Niveditha has completed B.E in Comp. Sci. Engineering in PB college of Engineering and M.Tech Information Security and Cyber forensics in Dr. M.G.R Educational and Research Institute. Currently she is pursing research in Dr. M.G.R Educational and Research Institute. She also published three papers in International journals and six papers in National conferences

Dr. T.V. Ananthan is currently working as Professor in Dept. of Comp. Sci. & Eng. and Inf. tech., Dr. M.G.R. Educational and Research Institute, Chennai. He has 26 years of teaching experience. He completed his Doctorate of Philosophy in the year 2012. He has published more than 40 papers in various International and National Journals and Conferences. His area of research is Networking, Wireless Communication, Mobile Communication and Network Security. He is a member in IEEE and IEI. He is also playing the role of reviewer in International Journals.

Dr.N.KANYA , Completed her Master of Science from Alagappa university in the year 2004 then completed Master of Technology in Computer Science and Engineering from Dr.M.G.R Educational and Research Institute University in the year 2007. She has completed her Ph.D in Computer Science and Engineering form *Manonmaniam Sundaranar University* in the year 2017. She has published totally 54 research articles. In that 17 research articles in International Journals, 12 research articles in International Conference and 25 research articles in National Conference. She has one Patent for “ Smart Helmet”. She is Life time member in “Quality Circle Forum of India” and also Member in IET, IAENG and SDIWC. With 20 Years of Teaching and Industry experience, she is currently working as Professor in department of Computer Science and Engineering at Dr.M.G.R Educational and Research Institute. She also designated as Internal Quality Assurance Cell In-Charge of the University.