Permission Based Android Malware Detection

Sachin Dighe, Akshay Jondhale, D.J. Salunke
Department Of Information Technology, Pravara Rural Engineering Collage,Loni, Maharashtra.

Abstract – Android mobile devices have arrive a widespread use since the past few years, thus leading to a developing in the number and variety of applications on the business market. However, from the overview of information security, the user control of responsive information has been shield by the quick development and rich variety of the applications. In the recent condition of the art, users are subject to responding plentiful requests for permission about using their personal data to be able execute an application. The consciousness of the user about data conservation and its relationship to permission requests is crucial for securing the user against defected software. Still, the slow transformation of users to novel technologies suggests the necessity for developing self-started tools for detecting defected software. In the present study, we analyze two main aspects of permission-based malware detection in Android applications: character selection methods and distribution algorithms. Within the framework of the assumptions specified for the study and the data used for the analysis, our findings reveal a greater performance for the Random Forest and J48 decision tree classification algorithms for most of the chooses details selection methods.

Index Terms – cyber security, android application, machine learning, static analysis, feature selection, classification, malware detection.

1. INTRODUCTION

Mobile contrivances of separate operating systems have presented a steep increase in the gone decade, thus popular to an incrimination in the number and variety of applications that run on mobile contrivances. Smart phones are recently used either in a parallel manner to a desktop computer or even to supersede it. A most proximate visually examine the orchestration of utilizing mobile phones reveal that they are mostly utilized for web browsing, convivial networking, and online banking. In advisement, they are utilized for mobile-categorical functions such as SMS messaging, read-time broadcasting of location, and all over access. As the efficiency in mobile phone functionality increase mobile phones become more and more fascinating for an immensely colossal population. Market data surveys reveal that the number of Smartphone sales ecumenical reached 208 million in 2012.

The evaluation of smart phones and mobile applications has conclude in a higher vicissitude in people's way of execution tasks in different aspects of daily life, such as engendering business and operate convivial communication. Mobile phone applications perform a higher variety not only in mundane daily life activities but adventitiously for users with more categorical use. From games to multimedia applications, navigation systems, and health-cognate applications, incipiently available mobile application markets such as Google's Play Store, Apple's App Store, or Microsoft's Windows Store offer a wide authoritative ordinance of applications to users with different demand [4]. The major application markets have been developing regularity both in terms of the applications sanction to the users and the downloads consummate by the users (e.g., visually perceive [5] for the evaluate of Goggle's Play Market since the past several years).

The rapid increase in the reputation of perspicacious phones has get to an amend in their possible as a target for malevolent activities [1]. This is mainly because users arrange access for different types of individual information by style of mobile applications [3]. As a conclude, some applications in the market have been relegate as developing malevolent activities. The diffusion of malevolent software is adventitiously moved by the policy of the market providers. For initially, Apple's App Store recently applies a policy that is subject to straight registration and company-expressed digital certification afore the turnout of any application, thus providing a security check for the applications listed in their application staging customarily [6]. Others, such as Google's Play Market, propose more immunity to developers in uploading their own software to the market. The applications are then ejected from the market in case of reported malevolent activity. In congruous, the Android operating system (in its recent form) sanctions supersession of the malevolent application from the contrivance casually. A same mechanism of ejected is withal employed by Apple's App Store (namely, by kill switch) [7]. The policy of post-find abstraction of malignant software brings the desideratum for early detection of malware applications. In the instant study, we purport on malware detection in Android operating systems. Malware detection study of Android applications may be implement in two ways: Static analysis and dynamic analysis [8]. Static analysis is conducted by reversion- engineering an application, without running the application. In particular, this process involves study the file with the .apk extension and by fixating on the content of the two files: Android Manifest.xml and classes.dex [9]. On the other hand, dynamic analysis (adscititiously called demeanor-predicated detection) is handle by working the application and study its execution traces in a controlled environment [10]. A distinction between the static analysis methods and the dynamic analysis methods reveals that the latter requires more computing resources in terms of recollection space and execution time.

This is due to the fact that in dynamic analysis, the traces are derive online during the course of applications working on the system, whereas online detection is not compulsory in static analysis [1]. The static analysis methods, however, require an intensive analysis of feature cull methods and processing algorithms for malware detection, as we explicate in more detail below. Android applications run by employing actions provided by the operating system. For this, applications need sundry sanctions that should be granted by the mobile contrivance utilize. Those requests for sanctions are customarily asked to the utilize during the course of installation of an application. The list of sanctions that may be asked to the utilize by Android application is provided by [9].

2. RELATED WORK

Earlier research employed many classification algorithms for the classification of malicious Android applications by static study. For detail, [3] performed a permission-based analysis of the applications at Google's Play Market by deriving the permission group from the relevant Android Manifest. xml files, as stated in the previous part. applying the Information Gain feature selection method to decrease the size of the feature set, and applying the J48 Decision Tree, Classification and Regression Tree (CART) and Random Forest classification algorithms onto the reduced data set, fulfillments were compared in terms of true positive, false positive, precision and recall rates. depending to their outcomes, J48 and Random Forest outperform CART by display higher precision rates and small false positive rates. In relative analysis, the behavior of choosed Android applications were organized by a feature-based learning framework which contain API calls as features combined with permissions.

The study employed SVM, J48 and Bagging classification algorithms. The outcomes shows that the combination of API calls with the requests for permissions increases the capability of malware detection [4]. In other study [1], numerous combinations of request for permissions, intents, broadcast receivers and original code were added to the possible set. The study was achieve by employing Random Forest classification and by using Android Malware Genome Project universal data, likewise a group of benign applications gained from third party markets. The outcomes revealed that the method outperformed conventional de-malware implements, even for novel malware software program. Other study [1] utilizing the Malware Genome Project data distribution Android applications with Bayesian classifier depend on aeonian code study. API calls, Linux system orders and sanctions were derived as features by reverse engineering the .apk files. Mutual Information (MI)maximization approach was acclimated to cull features containing classes of software applications (as defected or benign). outcomes reported better detection rates than then famous signature-predicated antivirus software program on the same group of malware specimen. In [3], a method was presented by analyzing the information only in disclosed files with J48 Decision Tree algorithm to find Android viruses. As introduced in Section I, malware detection methods do not only include static studies but additionally contain non-static analysis. Among many others, one approach, called Crow droid, employed crowd-sourcing for accessing application implementing traces.

The analysis applied a detectable framework to analyze Trojan horses from benign applications which had the same name and the same version but a different non-static posture. By uploading the crowd-sourcing application to Google's Play Market and giving the algorithm by public (but demeanor-cognate) data gained from the users, an efficient malware detection technical was design. The algorithms was depend on the system calls that built feature vectors for k-betokens relegation [10]. Another host-predicated behavioral analysis structure is called Andromaly, which perpetually monitors features and events obtained from the mobile agent. Andromaly then applies sundry relegation algorithms, containing k-betokens, logistic regression, histogram, decision tree, Bayesian networks and Naïve Bayes after feature cull (Chi-square, Fisher Score and Information Gain as feature cull algorithms) [7].

A. Data

The dataset, which was used in the study, was arranged by COMODO Security Solutions, Inc., a private security association [8]. A department of COMODO Security Solutions, Inc. is present at the center East Technical University (METU) campus, Turkey. The data set contains3,784 android applications, 2,338 of which are benign and 1,446 of which are malware applications. The stamp as 'benign' or 'malware' was provided in the data set, as declared by the company. This stamp was used for supervised learning in the feature selection and classification studies, as shown in the following section.

B. Methodology

Introduced in Section II, details selection allows ejecting some features from the data set which are previously unnecessary or irrelevant for the analysis. The concluding data set usually leads to a decreed processing period and greater accuracy compared to the basic dataset. In the current study, four feature selection methods and five classification algorithms were tested by inspecting the correctness of classification forecast of the applications into 'friendly ' or 'malware' . WEKA Analysis software tool was employed for evaluating the feature selection methods and the classification algorithms [9]. The development of the feature selection methods and the classification algorithms were performed by using the following measures.

Overall Accuracy (ACC)
True Positive (TP) Rate

False Positive (FP) Rate
Precision

C. Feature Extraction

In order to reach the sanction data of an application, the apk file must be extracted from the .apk elongated file and then the AndroidManifest.xml file must be read. We employed the APKTOOL [2] to extract the apk file into the Android Manifest file and the Smali file. Then, the sanctions defined in the Android manifest file were examined to extract which type of sanctions were required to run the application. Figure 2 depicts the processor apk file extraction.

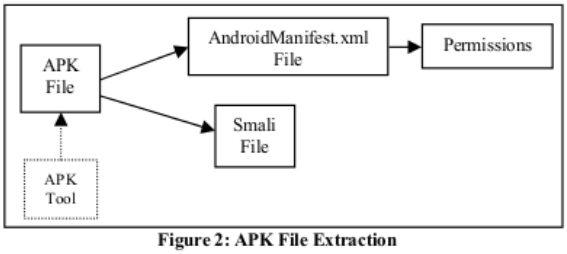

Figure 2: APK File Extraction

D. Feature Selection

Feature cull is performed to decrease the dataset size by abstracting the features (attributes) which are not propitious to be utilized in the analysis. The features are culled according to their representation capability of all the dataset. Efficient feature cull methods introduce performance gains by reducing the dataset size and the time spent in relegation analysis. Feature cull methods can be divided into two approaches, called attribute-predicated feature cull methods and subset-predicated feature cull methods. Attribute-predicated feature cull methods evaluate each feature discretely, independent of other features. It is mainly predicated on class features. Dependencies among the features are avoided, but the cognation to the class feature is taken into account. On the other hand, in subset-predicated feature cull methods, feature subsets are constructed desultorily and the subset that best represents the whole features is culled. Dependencies among the features are taken into account.

The steps below are followed for the cross-validation process:-

• **Step 1:** Divide the data randomly into k folds (subsets) of equal size.
• **Step 2:** Train the model on kí1 folds, use one fold for testing.
• **Step 3:** Repeat this process k times so that all the folds are used for testing.
• **Step 4:** Compute the average performance on the k test sets.

3. EXPERIMENTS

A. The evaluation of the relegation algorithm implementations were performed in terms of the following evaluation measures: Overall Precision, True Positive Rate, Erroneous Positive Rate, and Precision. These quantifications are derived from four rudimentary measures that are described below.

True Positive (TP): This quantification designates the number of correctly identified benign applications. Mendacious

Positive (FP): This quantification designates the number of incorrectly identified malware applications. In other words, a mendacious positive relegation identifies a pristinely maleficent application incorrectly as a benign application.

True Negative (TN): This quantification designates the number of correctly identified malware applications.

Erroneous Negative (FN): This quantification designates the number of incorrectly identified benign applications. In other words, erroneous negative relegation identifies a pristinely benign application incorrectly as a malevolent application.

Overall Accuracy: The ratio of correctly identified applications. In other words, it shows the ratio of correctly classified instances.

Overall Accuracy = (TP + TN) / (TP + TN + FP + FN)

True Positive Rate: The ratio of correctly identified benign applications. True Positive Rate is also called Recall.

TPR = TP / (TP + FN)

False Positive Rate: The ratio of incorrectly identified malicious applications.

FPR = FP / (TN + FP)

Precision: It is the ratio of retrieved instances that are relevant. It is also called positive predictive value.

Precision = TP / (TP + FP)

Revision reflects the fraction of correctly relegated instances within all relegated instances. Precision is a different measure than the precision of relegation. In the following section, we present the performance evaluation of the culled feature cull methods.

B. The Performance Evaluation of Relegation Algorithms

The Arbitrary Forest relegation algorithm exhibited the best performance virtually in all feature cull methods, except for the Relief Attribute Evaluator (25), for which the SMO relegation algorithm returned better precision, and the Gain Ratio Attribute Evaluator (25), for which theJ48 Decision Tree relegation algorithm returned better precision. Moreover, the J48 Decision Tree algorithm waste second best because it showed the best results for one(Gain Ratio Attribute Evaluator (25)) and the second best results for two feature cull methods (Gain Ratio Attribute Evaluator (25) and Cfs Subset Evaluator (25)).

The SMO relegation algorithm returned the second best results for two feature cull methods, consequently it can be evaluated as being worse when it is compared to the J48Decision Tree algorithm. The Bayesian relegation algorithm pointed out the most impuissant performance proximately for all feature cull methods, except for the Consistency Subset Evaluator (36).The results obtained by the evaluation of there legation algorithms suggest that the Desultory Forestland the J48 Decision Tree relegation algorithms might be preferred for the task of sanction-predicated Android malware detection. On the other hand, the results suggest that the Bayesian algorithm should be evaded in sanction predicated relegation analysis of Android applications.

C. The Compatibility of Feature Cull Methods and Relegation Algorithms

Bayesian relegation showed a relatively high-performance (the overall precision of 89.32%.) with the CfsSubset Evaluator with 25 features. The J48 Decision Tree algorithm returned high precision when it was with Gain Ratio Attribute Evaluator with 50 features and Cfs Subset Evaluator with 25 features, with 93.90% overall precision in both cases. The Arbitrary Forest showed the best performance with the Cfs Subset evaluator with 25 features and Gain Ratio attribute evaluator with 50 features, with94.90% and 94.50% overall accuracies respectively. This finding shows that 25 features and 50 features exhibited close precision. These were the highest overall precision values obtained in the analysis. The Relegation and Regression Tree (CART) algorithm exhibited the best performance with the Cfs Subset Evaluator with 25 features and Gain Ratio Attribute Evaluator with 50 features. Determinately, the Sequential Minimal Optimization (SMO)revealed the best compatibility with the Relief Attribute Evaluator with both 25 and 50 features.

4. CONCLUSION

In the present study, we performed a permission-based analysis of malware classification for Android applications in two major steps. In the first step, we applied a set of feature selection methods (Gain Ratio Attribute Evaluator,ReliefF Attribute Evaluator, Cfs Subset Evaluator and Consistency Subset Evaluator) to reduce the size of the feature dimension of the dataset, which originally had 182features. We then used the reduced datasets as input sets to five classification algorithms (J48 Decision Tree, Bayesian Classification, Random Forest, Classification and Regression Tree and SMO). The performance of the algorithms was evaluated by means of a set of measures, in particular by Overall Accuracy, True Positive Rate, False Positive Rate and Precision measures. The findings revealed that the Cfs Subset Evaluator feature selection method gave a good performance when it was used with 25 features. Moreover, increasing the number of features to 50 did not improve accuracy. As for the classification algorithms, the results suggested that the permission based classification analysis of Android applications can be more accurately performed with Random Forest and J48 Decision Tree classification but not with the Bayesian algorithm.

The prediction results of the classification algorithms revealed that the permission based Android malware detection model can be performed more accurately by using the Random Forest algorithm. Furthermore, the Random Forest and J48 Decision Tree work best with the Gain Ratio Attribute Evaluator with 50features and with the Cfs Subset Evaluator with 25 features, whereas Bayesian returned poor performance independent of the feature selection method. We expect that the results of the current study might provide the basis for future research in malware detection. Our future research will address a finer-grained analysis of the effect of dataset size in terms of the number of data points (i.e., the number of Android applications). We predict that an optimum dataset size can be determined depending on the selected feature selection method and the classification algorithm. In addition, the future research should address other static features existing in an apk files to conduct static analysis of Android malware, including but not limited to specific API calls though the requirements of higher resources for the analysis. Finally, the future research may focus on clustering analysis to identify emergent clusters in the dataset.

REFERENCES

[1] P. Felt, M. Finifter, E. Chin, S. Hanna and D. Wagner, «A Survey of Mobile Malware in the Wild,» In SPSM '11 Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, p. 3-14, 2011, Chicago, Illinois.

[2] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez and J. Blasco, «DENDROID: A text mining approach to analyzing and classifying code,» Expert Systems with Applications , 41(4/1), pp. 1104-1117, 2014

[3] Rapid7 Corporate Headquarters, «Mobile Security Guide: Protect Your Organization From Mobile Malware,» Rapid7 Corporate Headquarters, Boston, 2013.

[4] Symantec, «Securing the Mobile App Market: How Code Signing Can Bolster Security for Mobile Applications,» Symantec, 2012.

[5] Statista-The Statistics Portal,2014.\Available: http://www.statista.com/statistics/281106/number-of-android-app-downloads-from-google-play/.

[6] Symantec, «A Window Into Mobile Device Security: Examining the security approaches employed in Apple's iOS,» Symantec, 2011.

[7] Teufl, P., Kraxberger, S., Orthacker, C., Lackner, G., Gissing, M., Marsalek, A., Leibetseder, J., and Prevenhueber, O. (2012). «Android Market Analysis With Activation Patterns,» In Security and Privacy in Mobile Information and Communication Systems (pp. 1-12). Springer Berlin Heidelberg.

[8] Amamra, A., Talhi, C., and Robert, J. (2012, October). «Smartphone malware detection: From a survey towards taxonomy, » In Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on (pp. 79-86). IEEE.

[9] Tchakounté, F., & Dayang, P., «System Calls Analysis of Malwares on Android, » International Journal of Science and Technology, 2(9), pp. 669-674, 2013.

[10] Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011, October). «Crowdroid: behavior-based malware detection system for android, » In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (pp. 15-26). ACM.