Analysis on Malware Detection with Multi Classifiers on M0Droid and DroidScreening Datasets

Kyaw Naing Tun
ICT Department,
Central Institute of Civil Services (Lower Myanmar)
kyawnaingtun.ict@gmail.com

Zin May Aye
University of Computer Studies, Yangon
zinmayaye@ucsy.edu.mm

Kyaw Thet Khaing
Ministry of Education,
Nay Pyi Taw
kyawthetkhaing@moe-st.gov.mm

Abstract

The number of applications for smart mobile devices is steadily growing with the continuous increase in the utilization of these devices. the Installation of malicious applications on smart devices often arises the security vulnerabilities such as seizure of personal information or the use of smart devices in accordance with different purposes by cyber criminals. Therefore, the number of studies in order to identify malware for mobile platforms has increased in recent years. In this study, permission-based model is used to detect the malicious applications on Android which is one of the most widely used mobile operating system. M0Droid and DroidScreening data sets have been analyzed using the Android application package files and permission-based features extracted from these files. In our work, permission-based model which applied previously across different data sets investigated to M0Droid and DroidScreening datasets and the experimental results has been expanded. While obtaining results, feature set analyzed using different classification techniques. The results show that permission-based model is successful on M0Droid and DroidScreening data sets and Random Forests outperforms another method. When compared to M0Droid system model, it is obtained much better conclusions depend on success rate. Our approach provides a method for automated static code analysis and malware detection with high accuracy and reduces smartphone malware analysis time.

keywords: Mobile Malware Detection, Permission data, Classification techniques, M0Droid, DroidScreening.

1. Introduction

In recent years, smart mobile devices such as mobile phones, tablets, personal digital assistants (PDAs) have all become popular because of their processing capacity, smaller size and simpler design. Considering the fact that there are about 30 millions of smart phone users in Turkey, as soon as operating systems are concerned, the use of Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013.

As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system, and as of June 2018, the Google Play store features over 3.3 million apps.[1]. With an increased use of Android-based mobile devices, the number of malwares targeting these devices is also increasing day by day. Malwares which are embedded into the applications downloaded to mobile devices by means of internet may result in the device being captured by a cybercriminal, rendered dysfunctional or the privacy such as credit card information being stolen. The processes of detecting these malwares that normally cannot be recognized by users are usually performed by two different methods, namely static and dynamic methods.

In static analyzing approach, the behavior of the application is tried to be perceived via resource codes form the applications, and then, malwares are detected using these behaviors. According to the dynamic analysis approach, application analyses are performed based on their operability on mobile devices [2]. In this analysis, it is possible to detect the behaviors that cannot normally be detected by static analysis; however, running of the malware in the device poses a security thread for the mobile device. Therefore, in this study, by using the static analysis approach, it is accessed to manifest files of Android applications present in M0Droid and DroidScreening dataset which were previously analyzed and the relevant malwares the device are detected with permission data from these files. During the testing stage of system, different classification methods are used, and the permission-based attributes used are analyzed.

DroidScreening [3] framework scans applications at runtime to identify malicious patterns

of execution. The solution followed the standard objectives of the antivirus industry, that are the malware sample screening and the identification of threat level. DroidScreening employed both static analysis to extract key identifiers of the targeted Android application to train the machine learning model and lazy associative classification algorithms to produce the classification model. The novelty of the solution lays on the trigger dynamic execution-based analysis environment that generates various system-wide events to trigger malicious activity.

Their dataset numbered 1554 malware and 2446 benign applications. DroidScreening score danF1 value of 91.1%. In addition to F1score, the author scarfed a metric for the average misclassification cost, because they state that the cost of different misclassification errors should not have the same effect on the final accuracy result.

The remaining parts of the present paper are designed as follows: The related works are discussed in Section 2. The methods related to pre-processing, feature extraction and classification are explained in Section 3. Information about datasets is included in Section 4. In Section 5, experimental results are provided with an assessment of the best classification method. In the final Section, the results are summarized with referring suggestions.

The growing popularity of smart-phones and tablet computers has made mobile platforms a prime target for attack. The escalating number of users, however, also provides hackers with the opportunity to develop various malicious applications. In 2016, Kaspersky Lab detected 8,526,221 malicious installation packages, 128,886 mobile banking Trojans and 261,214 mobile ransomware Trojans.

2. Related Work

Detection of malicious application is achieved via attributes derived from the application file. In this context, the attributes are obtained by two different approaches in the literature, namely the static and dynamic analysis. In the system using dynamic analysis approach, extraction of attributes is achieved by using the fingerprint-based detection engine [4]. These attributes include permission data extracted from the manifest file, and semantics in byte code. It also includes data obtained from heuristic based detection engine called root privileges which controls the running application. MAST, a mobile application security tool, based on static analyzing approach, helps to bring out the behaviors that are likely to be malicious using limited analyzing resources. In this tool, Multiple Correspondence Analysis was used, which statically measure the relations among multi-categorical data during the feature extraction from application package. Code of practice and naturally presence of codes as well as permission data were used in developing this tool. The MAST tool was tested and compared by 5 different attribute groups and separate machine learning methods on 182 attributes [5]. In a method called DREBIN, the authors statically collected various attributes among Android applications, and they applied the support vector machines (SVM) on the resulting model. Along with permission data, they also used many attributes such as API calls, network addresses, hardware components, and gained a success rate of 93.9% with the SVM method [6]. Another study using the similar features is called DroidMat system, developed by Wu et al [7].

This system used permission data, API calls, constituents and instant messaging transitions as attributes in order to characterize the behavior of Android application by using static analyzing method. The modeling capacity of malware was attempted to be increased by applying k-means clustering method to these features. The number of clusters was determined by Singular Value Decomposition (SVD) method. Following the clustering process, 97% of success was achieved with k-NN classifier [7].

In this tool, Multiple Correspondence Analysis was used, which statically measure the relations among multi-categorical data during the feature extraction from application package. Code of practice and naturally presence of codes as well as permission data were used in developing this tool. The MAST tool was tested and compared by 5 different attribute groups and separate machine learning methods on 182 attributes [5]. In a method called DREBIN, the authors statically collected various attributes among Android applications, and they applied the support vector machines (SVM) on the resulting model. Along with permission data, they also used many attributes such as API calls, network addresses, hardware components, and gained a success rate of 93.9% with the SVM method [6]. Another study using the similar features is called DroidMat system, developed by Wu et al [7].

This system used permission data, API calls, constituents and instant messaging transitions as attributes in order to characterize the behavior of Android application by using static analyzing method. The modeling capacity of malware was

attempted to be increased by applying k-means clustering method to these features. The number of clusters was determined by Singular Value Decomposition (SVD) method. Following the clustering process, 97% of success was achieved with k-NN classifier [7].

Aafer et al. in [8] and Xiangyu et al. in [15] performed frequency analysis technique of the extracted API calls. In [8], the feature set was refined to use only API calls invoked by third-party applications. The APIs came mostly out of advertisement, web tracking, and web analysis packages. As the next step, they clustered the APIs by invocation methods (application specific, third-party package, or both). Their dataset consisted of 20,000 applications, 19.9% of which were malicious, and they were able to achieve 97.8% accuracy using machine learning algorithms. As a conclusion, they present statistics to argue in favor of the superiority of the API-based performance as compared to permission models. The same motif was engaged in [15] but produced opposite results. In this paper, the permission-based model managed to achieve 84.4% accuracy with the help of machine learning classifiers in contrast with the 70% that was achieved with the use of API-based model.

The permission data is of great importance in determining whether Android applications are malicious or not. In several studies, the permission data has been observed to significantly affect the success of classification as a common feature, even though different attributes were used. By only using permission data, Egemen et al. achieved a high performance in detection of malwares with a success rate of 98% using the Random Forest method [8]. In another study using permission data, permission based Android malware detection system, namely APK Auditor, was composed of three main constituents.

The systemconsists of a signature database for storing the attributes, a client for users requesting analyses, a database and a server for communicating to the server. By using this system, a total of 8762 applications were tested with a success rate of 88% [9].

3. Methodology and Datasets

We created four experiments, using both machine learning approaches on two analysis methods: permission-based clustering, permission-based classification, source code based clustering, and source code based classification. For training and testing of our machine learning models, we utilized M0Droid dataset, which contains 200 malicious and 200 benign android apps.

3.1 Signature based Malware Detection

Signature based malware detection methods are commonly used by commercial antimalware products. This method extracts the semantic patterns and creates a unique signature [12]. A program is classified as a malware if its signature matches with existing malware families' signatures. The major drawback of signature based detection is that it can be easily circumvented by code obfuscation because it can only identify the existing malwares and fails against the unseen variants of malwares. It needs immediate update of malware variants as they are detected.

Limitation of Signature Based Detection: Although signature based detection is very efficient for known malwares but it cannot detect the unknown malware types. Also because of limited signature database most of the malwares remain undetected.

3.2 Permission Based Detection

In Android system, permissions requested by the app plays a vital role in governing the access rights. By default, apps have no permission to access the user‟ data and affect the system security. During installation, user must allow the app to access all the resources requested by the app. Developers must mention the permissions requested for the resources in the AndroidManifest.xml file. But all declared permissions are not necessarily the required permissions for that specific application.

Limitation of Permission Based Detection: Permission based detection is a quick filter for the application scanning and identifying that whether the application is benign or malware but it only analyses the manifest file it do not analyze other files which contain the malicious code. Also there is very small difference in permissions used by the malicious and benign apps. Permission based methods require second pass to provide efficient malware detection.

3.3 Dalvik Bytecode Analysis

In Android, Dalvik is a register-based VM. Android apps are developed in java language, compiled in java bytecode and then translated to dalvik byte code. Bytecode analysis helps us to analyze the app behavior. Control and data flow

analysis detect the dangerous functionalities performed by malicious apps.

Limitations of Dalvik Bytecode Detection: In this method analysis is performed at instruction level and consumes more power and storage space. As the android devices are resource poor so they limits this detection approach.

3.4 Dataset

The author provides M0Droid dataset based on observing 1330 Malicious apk samples and 407 benign samples13. The original dataset had a total of 32342 data (feature vector) samples with 7535 benign samples (classified as positive class) and 24807 malicious samples (classified as negative class). In order to balance the two classes, we used the SMOTE package from Weka, and applied the Random filter to randomize the distribution of the two classes in the data file used. SMOTE generates data points of the under sampled class. Randomization was required as SMOTE adds data entries to the end of the data file. This would be a problem during cross validation when the folds are computed as we will have points primarily from the under sampled class in the last few folds. Performing this action provided us 48919 data samples with 21,112 samples of the positive class and 24807 samples of the negative class. For our experimentation we used this new sample.

There are two main approaches to analyze the Android malwares: Static and Dynamic Approach. Static approach is a way to check functionalities and maliciousness of an application by disassembling and analyzing its source code, without executing the application. It is useful for finding malicious behaviors that may not operate until the particular condition occurs.

Change Wifi State	Internet	Blue tooth	Send Stas	Sample Category
0.0	0.0	0.0	0.0	Bening
1.0	1.0	1.0	0.0	Bening
1.0	1.0	0.0	0.0	Bening
0.0	1.0	0.0	0.0	Malware
0.0	1.0	0.0	1.0	Malware
0.0	1.0	0.0	1.0	Malware
0.0	0.0	0.0	1.0	Bening
1.0	1.0	1.0	1.0	Malware

Table 1. A Small Part of Generated Dataset

4. Proposed System

To detect malicious software on a smartphone, this system uses M0Droid and DroidScreening data sets which enables determining typical indications of malicious activity and employs a broad static analysis that extracts feature sets from different sources and analyzes these in machine learning as described the following figure 1.

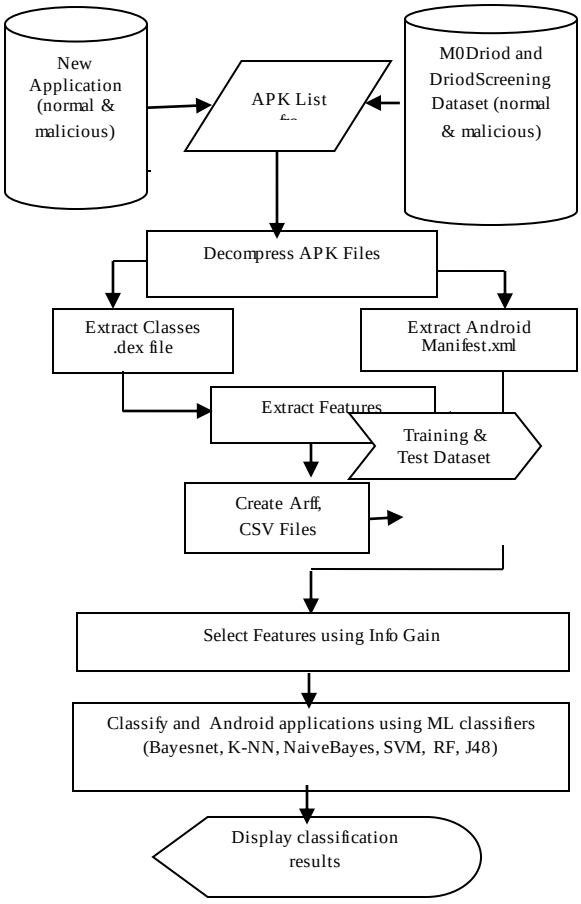

Figure 1: Proposed Framework on Android Malware Detection

This process outlined in the following:

a) Broad static analysis. In the frst step, M0Driod and DriodScreening are statically inspects a given Android application and extracts different feature sets from the application's manifest and dex code.

b) Embedding in vector space. The extracted feature sets are then mapped to a joint vector space, where patterns and combinations of the features can be analyzed.

c) Learning-based detection. The embedding of the feature sets enable to identify malware using effcient

techniques of machine learning, such as Random Forest, K-Nearest Neighbor and Support Vector Machines.

d) Explanation. In the last step, features contributing to the detection of a malicious application are identifed and presented to the user for explaining the detection process.

4.1. Machine Learning Classifiers

There are a large number of classifiers. Classifiers broadly break down into two classes: linear and non-linear. Linear algorithms attempt to separate n-dimensional data points by a hyperplane points on one side of the plane are of class X and points on the other side of class Y. Nonlinear classifiers, however, have no such restrictions; any operation to derive a classification can be applied. Unfortunately, this means that the amount of computation to classify a data point can be very high.

KNN is an non parametric lazy learning algorithm. That is a pretty concise statement. When you say a technique is non parametric , it means that it does not make any assumptions on the underlying data distribution. This is pretty useful, as in the real world , most of the practical data does not obey the typical theoretical assumptions made (eg gaussian mixtures, linearly separable etc) . Non parametric algorithms like KNN come to the rescue here.

Random forest is a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges as to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them.

A Support Vector Machine as stated by Luis et al (SVM) performs classification by constructing an N-dimensional hyper plane that optimally separates the data into two categories. SVM models are closely related to neural networks. In fact, a SVM model using a sigmoid kernel function is equivalent to a two layer, perceptron neural network. Support Vector Machine (SVM) models are a close cousin to classical multilayer perceptron neural networks. Using a kernel function, SVM's are an alternative training method for polynomial, radial basis function and multi-layer perceptron classifiers in which the weights of the network are found by solving a quadratic programming problem with linear constraints.

5. Experimental Results

The evaluation of machine learning classification algorithm on permission based classification is presented in Table 1.

Algorithm	Precision	Recall	F-Measure
BayesNet	0.986	0.985	0.969
K-NN	0.998	0.998	0.998
NaiveBayes	0.957	0.957	0.957
SVM with smo	0.989	0.989	0.989
Random Forest	1.000	1.000	1.000
J48	0.999	0.999	0.999

Algorithm	Precision	Recall	F-Measure
BayesNet	0.959	0.958	0.958
K-NN	0.981	0.981	0.981
NaiveBayes	0.590	0.593	0.524
SVM with smo	0.782	0.780	0.780
Random Forest	0.990	0.990	0.990
J48	0.971	0.971	0.970

Table 2. Evaluation results of permission-based classification using single machine learning algorithms

As it can be observed from the table, support vector machines with sequential minimal optimization performed the best with F-measure of 0.780. This algorithm also correctly classified 78.0% of test instances in 10-fold cross validation. The algorithm is efficient in terms of speed. Only 0.04 seconds were needed to train model and instances are classifed very fast, so this approach is suitable for live classifcation of the applications. We have integrated this model for classifcation based on permissions with SVM in OWASP Seraphimdroid Android application that can be obtained from Google Play store. This proves that modern smartphones have enough computational power to perform machine learning classification based on the permissions. On the other hand, Bayesian algorithms such as Naive Bayes and Bayesian networks performed the worst. This may be due to the small dataset, containing only 387 instances. Bayesian algorithms usually require much more data than SVM in order to train the model with high accuracy. With more data, even SVM model may slightly improve. SVM algorithm performs better on statistical t-test with confidence interval of 0.05 than Naive Bayes, Bayesian Network, JRip and Logistic regression, however, it is statistically not significantly better than decision trees and random forests. In Table 2, we present the results of ensemble learning using majority voting. We experimented with the ensembles of three algorithms in order to see which

algorithms contribute to the best results in ensembles. Three algorithms that performed the best were SVM with SMO, Logistic regression and Random forest performing with an F-measure of 0.891 and classifying 89.1% correctly. This is only a slight improvement compared to SVM algorithm alone and t-test showed it is not significantly better with the confidence interval of 0.05. On the other hand, ensemble algorithms performed much slower. Obviously, they need more time to apply multiple, in our case three or five machine learning algorithms and then to post-process results. Since the significance test showed that the performance of the ensemble learning algorithm is not significantly better than single machine learning algorithm, there is no real sense of using these algorithms in production.

The file "AndroidManifest.xml" created by the developer of Android applications not only includes some application-related basic information such as the name of package and version, but also includes user permissions. Each application requests the related permissions in the manifest file from user.

Table 3: # of sample for M0Driod Dataset

Dataset	# of sample		
	Bening	Malware	Total
Training	7535	24807	32342
Testing	660	493	1153

Table 4. Top Extracted Features

Category	Features	Weight
API Calls	sendSms()	1.12
Permission	SEND_Sms	0.84
Permission	Change_Config	0.57
Permission	Read_Phone_State	0.50
API Calls	getSubscriberID	0.49
Sys Call Log	system/bin/su	0.44
API CAlls	HttpPost	0.38

6. Conclusion and Future Work

In this study, the detection of malwares for Android platform using static approach is performed on M0Droid data set. In the experiments, permission based data from the manifest files are used as attributes. The results are obtained using different classification methods, and the most common attributes between the categories are examined. we proposed an ML-based malware detection and classification methodology together with the application of static analysis on an extensive dataset of Android applications. To this end, we designed a tool, M0Driod and DroidScreening , to extract as many informative features as possible from our dataset. We considered mainly features from the Dalvik bytecode.

In the study of creating M0Droid data set, a success rate of 69.5% is achieved using API system calls [10]. In our study, however, malicious software are detected using permission data from the same data set. When examined the experimental results, it is observed that success above 90% could be achieved by means of other classification methods except Naive Bayes. In addition, malicious software were automatically detected with the highest success rate of 94.5% in our study. In this context, the result increases the success rate of reference by 25%. When examined the attributes and taking the results into account, it can be said that the permission-based data are more efficient than attributes obtained by system calls. The features extracted were converted into feature vectors, each containing 560 binary features. We then applied feature selection on the aforementioned extracted features, which led to the selection of 101 informative binary features suitable to feed our proposed classification methodology. Moreover, we performed an extensive Grid-search analysis along with a 10-fold Cross-validation to tune the hyper-parameters of the classification algorithm to maximize the prediction accuracy.

We performed Family-by Family classification and obtained an average accuracy score of 92% in classification of unseen malware. In addition to this, we conducted a cumulative classification in order to investigate how well old malware can contribute to the detection of new variants of both known and unknown (zero-day) malware. We achieved reasonable accuracy rate, hence proving the robustness of the features extracted. As future work, we will extend our Android application analysis and combine static analysis with feature sex tracted from dynamic analysis to compensate the limitation associated with static analysis and get the best of both static and dynamic analyses. We will extract more features related to the behavior of Android applications such as CPU and Memory consumption, Network traffic activities, Inter-Process Communications (IPC) and system calls made by applications so as to interact with Android OS. In addition to these, we will make use of classification

algorithms such as SVM along with different kernels to deal with structured data (e.g., string, set, graph, etc.).

In this work, an effective android malware detection system with machine learning approaches has presented for malicious Android applications. This system compared the accuracy results among machine learning classifiers. As the vast majority of mobile malware targets the Android platform, this work focuses on Android malware detection using machine learning approaches. By combining results from various classifiers, it can be a quick filter to classify more suspicious applications. However, the method presented can be adapted to other platforms with minor changes.

References

[1] https://en.wikipedia.org/wiki/ Android (operating system)

[2] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani,, "Crowdroid: Behavior-Based Malware Detection System for Android", SPSM '11 Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, pp. 15-26, 2011

[3] M. Sun, X. Li, J. C. S. Lui, R. T. B. Ma, Z. Liang, Monet: "A User-oriented Behaviour-based Malware Variants Detection System for Android", IEEE Transactions on Information Forensics and Security, 2017, pp. 1103-1112.

[4] Y. Zhou, Z. Wang,, W. Zhou, and X. Jiang, "Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets", 19th Network and Distrubted System Security Symposium (NDSS 2012), San Diego, CA, USA, 2012

[5] S. Chakradeo, B. Reaves, P. Traynor, and W. Enck, "MAST: Triage for Market-scale Mobil Malware Analysis", 6th WiSec, Budapest, Hungary, 2013.

[6] D. Arp, M. Spreitzenbarth, Hübner, H. Gascon,, and K. Rieck, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket", Network and Distributed System Secuirty Symposium, 2014

[7] D-J. Wu, C-H. Mao, T-E. Wei, H-M. Lee, K-P. Wu, "DroidMat: Android malware detection through manifest and API calls tracing", Information Security (Asia JCIS), Tokyo, pp. 62-69, 2012

[8] J. Yu, Q. Huang, and C. Yian, "DroidScreening: a practical framework for real-world Android malware analysis," Security and Communication Networks, vol.9, no.11, pp.1435–1449,2016.

[9] Y. Aafer, W. Du, and H. Yin, "DroidAPIMiner: mining API level features for robust malware detection in android," in Security and Privacy in Communication Networks, vol. 127 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 86–103, Springer,2013.

[10] N. Idika and A. P. Mathur, "A survey of malware detection techniques", Purdue University, p. 48, 2007.

[11] Xiangyu-Ju, "Android malware detection through permission and package, "in Proceedings of the 2014 International Conference on Wave let Analysis and Pattern Recognition, ICWAPR 2014, pp. 61–65, Lanzhou, China, July 2014.

[12] P. Faruki, V. Ganmoor, V. Laxmi, M. S. Gaur, and A. Bharmal, "AndroSimilar: Robust Statistical Feature Signature for Android Malware Detection," Proc. 6th Int. Conf. Secur. Inf. Networks, pp. 152–159, 2013.

[13] M. Zheng, M. Sun, and J. C. S. Lui, "DroidAnalytics : A Signature Based Analytic System to Collect , Extract , Analyze and Associate Android Malware," 2013.

[14] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," ACM Comput. Surv., vol. 44, no. 2, pp. 1–42, 2012.

[15] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," Proc. - 2010 Int. Conf. Broadband, Wirel. Comput. Commun. Appl. BWCCA 2010, pp. 297–300, 2010.

[16] "strace download | SourceForge.net." [Online]. Available:http://sourceforge.net/projects/strace/.[Accessed: 22-Dec-2015].

[17] M. Zhao, F. Ge, T. Zhang, and Z. Yuan, "AntiMalDroid: An efficient SVM-based malware detection framework for android," Commun. Comput. Inf. Sci., vol. 243 CCIS, pp. 158–166, 2011.

[18] A. Aiken, "Apposcopy : Semantics-Based Detection of Android Malware Through Static Analysis," Fse 2014, pp. 576–587, 2014.

[19] X. Jiang. Security alert: Golddream, 2011. http://www.csc.ncsu.edu/faculty/jiang/GoldDream/

[20] X. Jiang. Security alert: Gingermaster, 2011. http://www.csc.ncsu.edu/faculty/jiang/GingerMaster/

[21] X. Jiang. Security alert: New droidkungfu variant, 2011.www.csc.ncsu.edu/faculty/jiang/DroidKun gFu3

[22] W. Enck, M. Ongtang, and P. D. McDaniel. On lightweight mobile phone application certification. In Proc. of ACM Conference on Computer and Communications Security (CCS), pages 235–245, 2009.

[23] B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. NitaRotaru, and I. Molloy. Android permissions: a perspective combining risks and benefits. In Proc. of ACMsymposiumonAccessControlModelsandTechnologies (SACMAT), pages 13–22, 2012.

[24] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In Proc. of IEEE Symposium on Security and Privacy, pages 95–109, 2012.

[25] Chun-Ying Huang, Yi-Ting Tsai, and Chung-Han Hsu "Performance Evaluation on Permission-Based Detection for Android Malware"

[26] Daniel Arp1, Michael Spreitzenbarth2, Malte H¨ubner1, Hugo Gascon1, Konrad Rieck1, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket"