MANIFEST FILES CLASSIFICATION OF ANDROID MALWARE

Chit La Pyae Myo Hein
University of Computer Studies, Mandalay, Myanmar.

ABSTRACT

Malicious activities on mobile devices, e specially f or a ndroid devices are s preading a round t he world and i nfecting not only for end users but also for large organizations and s ervice pr oviders. T herefore malware an alyzing an d detecting methods a re pr oposed by m any researchers. Among many approaches for m alware d etecting, s tatic approach i s s till c onducting be cause the a dvantages of t his a pproach which c an r educe c ost, t ime a nd r isk rather t han d ynamic a pproach. T he main obj ective of our research w ork is t o pr opose a s tatic-based malware detection f ramework which s hould improve t he ove rall a ccuracy a nd performance o f t he malware d etection. Although t he pr oposed f ramework i s described i n t his pa per, the detail explanation a nd e valuation f or t he whole f ramework i s n ot i ncluded. This framework is th e c lassification of a ndroid a pplications ba sed on machine l earning t echniques. T he framework r ests on a c ombination of requested pe rmission of t he m anifest files a nalysis a nd s tatic approach. This pa per onl y emphasizes on t he classification of th e mo bile ma lware according t o t heir l evel of pr otection by u sing Support V ector M achine (SVM) algorithm. The ex perimented results o f c lassification f or ma lware are also discussed in this paper.

KEYWORDS

Classification, feature s election, Android, Malware, M anifest f iles, Mobile Security

INTRODUCTION

SMARTPHONE have b een a vulnerable t arget f or m alware s ince June 2004.T he num ber of infected applications s teadily in creased u ntil certain s ecurity m easures l ike application s igning and v alidation of developers w as i ntroduced. Android permits a pplication in stallation f rom third party vendors mean that Google has no c ontrol ove r t he qua lity o r safety o f the applications provided in these s tores. S everal cas es w ere encountered w here l egitimate ap plications f rom t he G oogle P lay S tore were mo dified to in ject ma licious code a nd t he m odified a pplications were s old i n these t hird pa rty s tores. It is difficult to determine whether the application i s g enuine or not . T he

reliability of the application depends upon the security measures implemented by the application store. The attacker can easily change the code in order to incorporate the malicious code and repackages the application and publishes them in the application market. Users usually cannot differentiate between the malware application and the legitimate and application and thereby end up installing by malwares.

OBJECTIVES OF THE STUDY

The main objective of our research work is to propose a static-based malware detection framework which should improve the overall accuracy and performance of the malware detection.

In this paper, we propose a malicious application detection framework on android market to solve above problems. This framework is able to perform both detection methods using self-organizing feature map on android market .So that the problem of static detection based on permission can be solved. Therefore, the permission based malware detection on Android Application using the Android Asset Packaging Tool (aapt) to extract and decrypt the data from the Android Manifest.xml file. Android Application is analyzed by using previous behaviors of malware and if it is suspicious application, it can be automatically detected.

RELATED WORKS

This section summarized the research papers concerning with malware detection. Static approach (code-based), dynamic approach (run time-based), and hybrid of static and dynamic approach are usually used to detect the malware in the previous research works.

Rieck et al. propose a framework for automatic analysis of malware behavior using machine learning. The framework allows for automatically identifying novel classes of malware with similar behavior (clustering) and assigning unknown malware to these discovered classes (classification). Zhou and Jiang collected and analyzed over 1200 malware samples for Android from 2010 to 2011. They found was the use of an update component that downloaded malicious code at runtime. These new techniques make it much harder to recognize an application as malware because most malware detection tools use static analysis or signature matching. If a malicious application is downloading the payload at runtime, traditional malware detection will not work. Barrera et al. use self-organizing maps to analyze permission usage patterns in applications. Felt et al. applied a type of static analysis to verify if an Android application is over-privileged or not. It examines all the permissions an application requests, and in case of not using requested permission, it concluded that the application is over-privileged. Burguera

et al. also applied the static approach to analyze the behavior of malware Android application. After analyzing, he proposed a model called Crowdroid which classify the malware or benign by applying machine learning methods. Enck et al. proposed a framework to detect potentially malicious applications based on permissions requested by Android applications. Static approach is used in their proposed framework.Enck et al. also presented a malware detecting framework called "TaintDroid" which is based on dynamic approach to monitor sensitive information on smartphones. Bläsing et al. used an Android Application Sandbox to perform Static and Dynamic analysis on Android applications. While static analysis scans android source code to detect malware patterns,dynamic analysis executes and monitors Android applications in a totally secure environment.

PERMISSION BASED ANDROID SECULITY MODEL

Android is free and based on a 2.6 Linux Kernel. Figure 1 shows the four layers of the Android operating system. The Linux kernel resides on the lowest layer. The second layer contains the native libraries of the Android operating system. The libraries can be used freely by any of above two layers. Application framework layer contains the Java frameworks which essentially represent the API that can be accessed by application developers. Application reside on the topmost layer can use the underlying frameworks and libraries. Many applications are preinstalled and provide the main functions of the devices such as telephony and SMS.

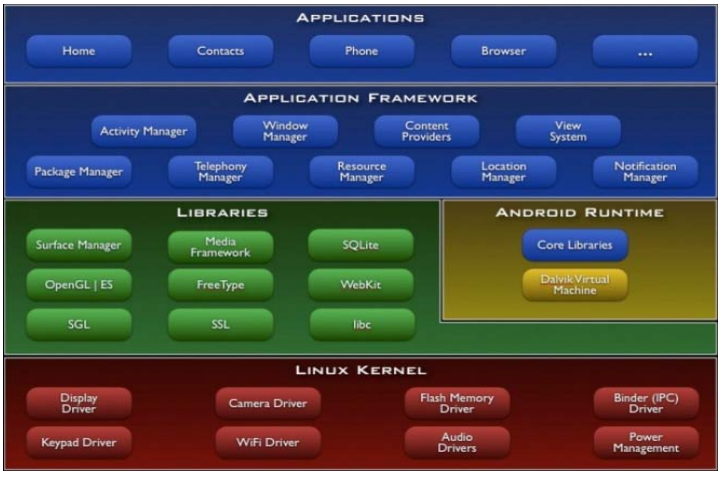


Figure 1. Architecture of Android.

A. Android Permission Model

At the core of the Android security model is a permission-based system that provides controlled access to various system resources. The expressiveness of the permission set plays an important role in providing the right level of granularity in access control.Android has a large number of permissions restricting access to advanced functionality on devices, only a small number of these permissions are actively used by developers.

By default, android permission system denies access to functionality or other applications that could negatively impact installed on the device. Examples of these functionalities are sending message or making phone calls (which may cost to the user); keeping the device screen on or accessing the vibrator (which may drain the battery); and reading the user's address book (which may violate the user's privacy).

To make use of the restricted functionality, android requires application developers to declare which of the restricted features are intended to be used by their application. There are 100 and over items of functionality which may be implicit (predefined permission) and explicitly (developer-defined permission).Some examples of permission to be requested the system resources in an android application are;

INTERNT	allows accessing the Internet
RECEIVE_SMS	for monitoring, recording or processing incoming SMS
RECORD_AUDIO	for recording audio message
READ_FRAMEBUFFER	directly reading the frame-buffer
SET_WALLPAPER	Allows applications to set the wallpaper.

B. Android Manifest File

Every android application composes in the form of package APK (apk) file format. After extracting the APK (apk) package, the following different types of file are also composed as shown in Figure 2. Among these files, an android *Manifest File with XML format* is one of the important file. Because, the manifest file presents essential information about the application to the Android system. Permissions are declared it th e manifest using the *permission* tag followed by a common namespace (android.permission) as described in Figure 3.

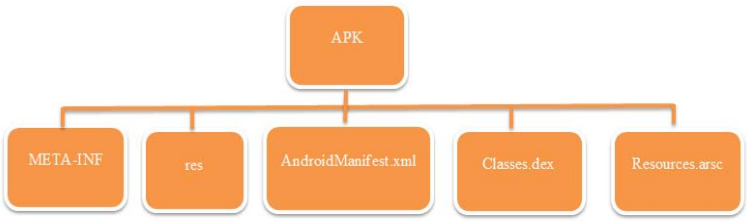


Figure 2. Files Containing in APK Package

Some information containing in manifest.xml file are described in Figure 3 as example.

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/
android">
<use-permission
Android:name=
"android.permission.RECEIVE_SMS"/>
Android:name= "android.permission.SENT_SMS"/>
</manifest>
```

Figure 3. Permission Information in Manifest.xml File

C. Permission Protection Level

Android system defines four protection levels, which characterize the potential risks implied in the permission and enforce different install-time approval processes. Permissions have associated protection levels:

Normal: They pose a low-risk factor and typically only affect the application's scope. Normal permissions are granted by the system automatically without explicit approval of the user.

Dangerous: They are higher-risk permissions that allow costly access to services. The permissions can be granted by the user during installation. If the permission request is denied, then the application is not installed.

Signature: permissions are only granted if the requesting application is signed by the same developer that defined the permission. Signature permissions are useful for restricting component access to a small set of applications trusted and controlled by the developer.

SignatureOrSystem: permissions are granted if the application meets the Signature requirement or if the application is installed in the system applications folder. Applications from the Android Market cannot be installed into the system applications folder. System applications must be pre-installed by the device manufacturer or manually installed by an advanced user.

PROPOSED FRAMEWORK

The first objective of our framework is to reduce the detecting and classification time for malware *by introducing the features selection and extraction step* in the framework. The second objective is to *classify* and *characterize* the malware by only taking the manifest file analysis in opposition to existing machine learning approach.

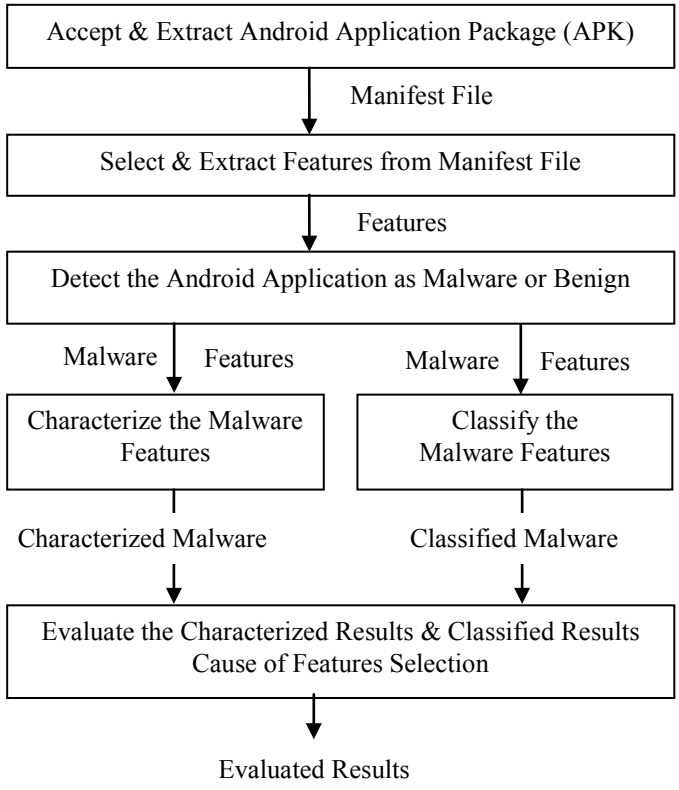


Figure 4. Proposed Malware Detecting Framework

The cost of time and risk for detecting malware can reduce by means of static approach rather than dynamic approach. Therefore this framework is also based on static (code-based) approach. The components in this framework are as follows:

Android Application File Accepting Component: This component is firstly responsible to accept the Android Application Package (APK) with (.apk) file format. Secondly the *manifest.xml* file is needed to extract from Android Application Package (APK).

Featuree Selection Components: The output of the first component (*manifest.xml*) is used as the input of second component. After that permission information, i.e., *features of application* must be extracted from manifest file.The correlated and more meaningful features should be selected by using or proposing the feature selection methods. (We are proposing and testing a feature selection method which is based on Manifest file analysis.)

Malware Detecting Component: This component is reponsible to detect the inputting android application is malware or goodware (benign) by using the features getting from second component. This can be done by proposing a new method which should be better than existing detecting method.

If the unknown application is detected as bengin, there is no need to proceed remaining phases. If the unknown application is detected as malware, the process of characterization or classification can be done as next step.

Malware Characterization Components: This component is only responsible to characterize the malware such as this malware is under the group of privacy concern, memory usage concern, system resources concern, etc. (We have to also propose a new characterization method by also using Manifest file analysis approach.)

Malware Classification Component: This component is responsible to classify the unknown malware as their types, names, security and risky level, permission level based on their training dataset. (We are proposing and testing a new classification approach which is based on malware feature pattern mining approach virsus with the existing classification approach.)

Before finishing the testing of our pattern mining approach for feature classification, the testing results on some classifiers are described in this paper.

EXPERIMENT FOR CLASSIFIER

Portions of experiment for different components have been making to evaluate the performance of the whole proposed framework. This paper only describes the experimental results concerning with the malware classification according to their permission levels.

Experimental Set Up: Known Android Malwares according to their permission levels (as described in (C)) are downloaded from Cantagio

(http://contagiodump.blogspot.com/). Similarly android benign applications are also downloaded from Android Market. There are totally 230 malwares and 86 benign applications are collected as the dataset. Two third(2/3) of this dataset are used as training dataset and one third (1/3)of this dataset are used as testing dataset for classification.

Feature Extraction: Manifest.xml files are extracted from two third of Android application packages (APK). Then application features (application's permission requests to system resources) are also extracted by using the Android Asset Packaging Tool (aapt). The sample application and permission feature matrix is described in Table 1.

Table 1. Sample Application & Permission Feature Matrix.

	a.p.ACCESS_WIFISTATE	a.p.BATTERY_STATE	a.p.READ_SMS	a.p.WRIRE_SMS	a.p.GET_ACCOUNT	a.p.CLEAR_CACHE	a.p.DELETE_SMS	a.p.FACTORY_TEST	Class Label
holycolbert.apk	0	0	1	1	0	0	0	0	Normal
AndroidDogowar.apk	0	0	1	1	0	1	1	0	Signature
Lovetrap.apk	0	0	1	1	0	0	0	0	Dangerous
BatteryDoctor.apk	0	1	1	1	0	0	1	0	Normal
htc.apk	1	1	1	0	1	1	0	0	Signature
MonkeyJump.apk	1	1	0	2	1	1	1	0	SigOrSys
Bgserv.apk	1	0	1	1	1	1	1	1	SigOrSys
hippo_sample.apk	0	0	0	1	0	1	0	1	Normal
SPPush_1314935990854.apk	0	0	0	0	1	0	1	0	Dangerous
jimm.apk	0	0	0	1	1	1	1	1	Dangerous

Classification Model: We have to build classification model as a part of our framework. We have employed two different algorithms for the classification: ID3 and SOM. The natures of these two algorithms are different from each other. The nature of ID3 algorithm is based supervised learning approach. Although actual nature of SOM is based on unsupervised learning approach, SOM is employed for classification purpose in some research works. One of our objective is to evaluate the strength of SOM for classifiers in opposed to other classifiers. Therefore the features dataset of android application is tested by using SOM and ID3 algorithms.

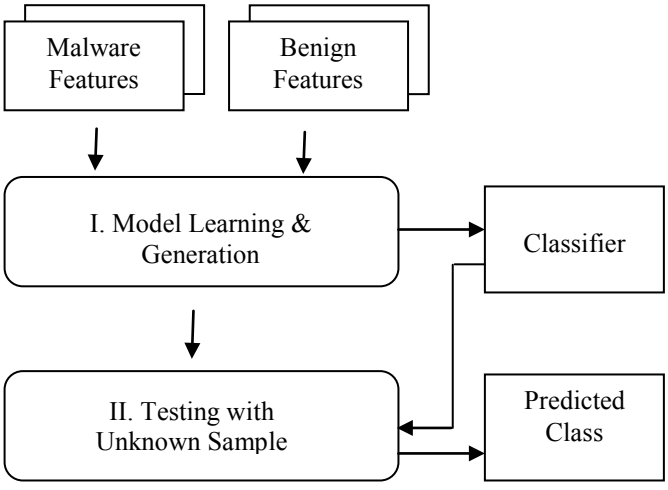


Figure 5. Classification Approach

There are two main phases in classification as described in Figure 5. The first one is model learning and classifier generation phase and the second one is testing phase with unknown dataset. The *classifier is generated* by the first phase and *classifier is used* by the second phase for testing.

We build the classification model based on the training set by using SOM and ID3 algorithms.

In the Self Organizing Map (SOM) method, the applied learning is an unsupervised learning where the network does not utilize the class membership of sample training, but use the information in a group of neurons to modify the local parameter. The SOM system is adaptively classify samples data into classes determined by selecting the winning neurons are competitive and the weights are modified.

The algorithm on the SOM

neural network as follows:

a. If the feature vector matrix of size k x m (k is the number of feature vector dimensions, and m is the number of data), the initialization:

- The number of the desired j class or cluster
- The number of component i of the feature vector matrix (k is the row of matrix)
- The number of vector Xm,i= amount of data(matrix column)
- The initial weights Wji were randomly with interval 0 to 1
- The initial learning rate α(0)
- The number of iteration (e epoch)

b. Execute the first iteration until the total iteration (epoch)

c. Calculate the vector permission to start from 1 to m:

$$D(j) = \sum_{i} (W_{ji} - X_{mi})^2$$

For all of j

- Then determine the minimum value of $D(j)$
- Make changes to the j weight with the minimum of $D(j)$

$$\sum_{W_{ji}}^{new} W_{ji} + \propto (X_{mi} - W_{ij})$$

d. Modify the learning rate for the next iteration:

$$\propto (t+1) = 0{,}5 \propto (t)$$

Which t start from the first iteration to e.

e. Test the termination condition

Iteration is stopped if the difference between Wji and Wji the previous iteration only a little or a change in weights just very small changes, then the iteration has reached convergence so that it can be stopped.

f. Use a weight of Wji that has been convergence to grouping feature vector for each malware, by calculating the distance vector with optimal weights.

g. Divide the malware features (Xm) into classes:

If $D(1)<D(2)<D(3)<D(4))$, then the malware protection included in normal .

If $D(2)<D(1)<D(3)<D(4)$, then the malware protection included in dangerous.

If $D(3)<D(2)<D(1)<D(4)$, then the malware protection included in signature.

If $D(4)<D(3)<D(2)<D(1)$, then the malware protection included in signatureorsystem.

Thus, applications can be assigned to the nearest neuron, effectively classifying of the applications requesting similar permission into the same neighborhood. To visualize the classify structure of high dimensional weight vectors of SOM neurons, a

graphic d isplay c alled U -matrix i s used. Describe t he p rocess u sed t o collect th e p ermission d atasets f rom the cantagio malware site and android market of t he 300 a pplications.The experimental 45 application results will b e s een f rom t he l arge percentage o f S OM a ccuracy i n cl assifying the m alware c orresponding t o t he class by u sing ma nifest f iles Fig. 6, performance r esults w ere b est achieved b y S OM protection l evel, based on the analysis of the tests and experimental r esults o f a ll th e 4 classifier.

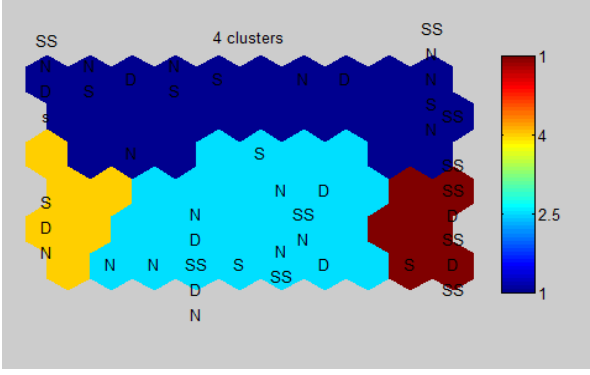
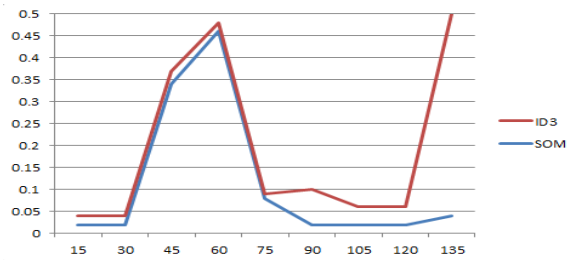
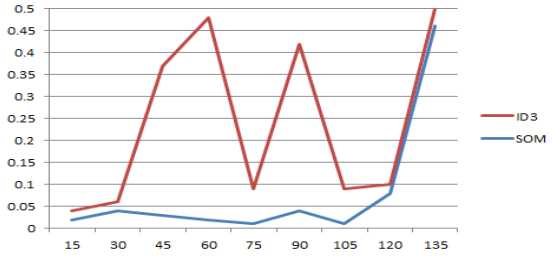


Figure 6. Classifier of android malware protection level in SOM

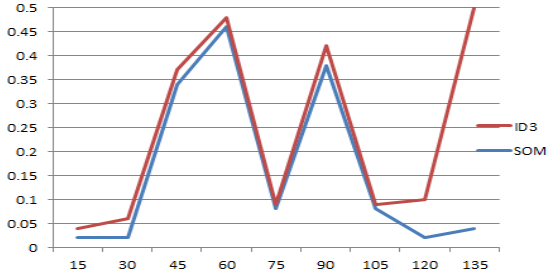
Machine l earning m ethods on mobile devices are important to select an a ppropriate m ethod d epending on the p articular application. ID3 decision tree m ethod is w ell s uited to o ur problem o f filtering large amounts of applications as i t can p erform r elatively f ast c lassification w ith lo w computational overhead once trained. Detecting s uspicious Android a pplications is the ability to model both an 'expert' a nd 'learning' system w ith relative ease co mpared t o ot her machine learning techniques. Moreover, this method is reducing the amount of features s hould be pe rformed w hile preserving a h igh l evel of ac curacy. ID3 depends on e ntropy of t he attributes a nd it s elects th e la rgest value o f g ain as t he b est f eature. Figures 7 depict for each experiment, the av erage A ccuracy a nd T PR,TNR results of the experiments undertaken to ev aluate t he classifier o f ID3 an d SOM. X-axis d ifferent sets o f features were used containing 15, 30, 45,60,75,90,105,120 and 135 features respectively and y -axis each o f t he android applications.



(a) TPR



(b) TNR



(c) Accuracy

Figure 7. Classifier performance comparison (ID3 and SOM)

The t esting i nstances a re f ed into t he m odel t o ev aluate e ach classifier's performance. *True positive ratio, true negative ration and overall model's accuracy* are measured t o evaluate the model's performance.

- **True Positive Ratio**: is th e proportion of malware i nstances t hat were correctly classified.

TPR=TP/TP+FN

- **True Negative Ratio**: is the proportion of be nign i nstances t hat were correctly classified.

TNR=TN/TN+FP

- **Overall Accuracy**: is t he to tal number of b enign a nd m alware instances co rrectly classified d ivided

by th e to tal number of t he da taset instances:

Total Accuracy:
(TP+TN / TP+FN+FP+TN)

CONCLUSION

In this paper, we proposed the new framework to obtain and analyze the p rotection le vel of ma lware detection. W e can use only manifest files to detect malware. Manifest files are r equired i n a ll A ndroid a pplications, a nd t hus, t he pr oposed method i s a pplicable t o a ll A ndroid applications. Static b ased m alware detection f ramework for a ndroid application using the Self-Organizing Map (S OM) a lgorithm. Malware analysis shows that in android a small number pe rmissions a re ve ry f requently us ed a nd a l arge num ber of permissions a re onl y oc casionally used. M oreover, m alware an alysis shows correlations between several of the infrequently us ed pe rmissions. Testing a nd t raining a pplications ar e then pa ssed t hrough t he be havior based m odule f or i dentification of android p ermission mo del. W e realize that pe rmission l evels of a ndroid a small num ber pe rmissions a re ve ry frequently used and a large number of permissions a re onl y oc casionally used an d s how t hat i t can ach ieve high accuracy rate.

Future work w ill e mphasize on t esting o f al ready t ested m alware applications t o d educe t he cl assification of t heir pe rformance, t esting of ot her m ajor a pplications i n the Android M arket, a nd ot her testing to discover a dditional m obile de vice vulnerabilities. T he analysis on m alicious and benign application to create a malware detection system based on unsupervised l earning, capable t o l earn a c oncept of nor mality and therefore capable to detect malware, which has been unknown until now.

BIBLIOGRAPHY

A. Shabtai, Y. Fledel, and Y. Elovici, Automated Static Code Analysis for Classifying Android Applications Using Machine Learning, International Conference on Computational Intelligence and Security,(2010).

A. Ultsch and H. Siemon. Kohonen's self-organizing feature maps for exploratory data analysis. In Proceedings of the International Neural Network Conference (INNC'90), Dordrecht, Netherlands, Kluwer,(1990).

Android apktool.https://code.google.com /p/android-apktool/

Android market API.http://code.google.com/p/a ndroid-market-api/

Android Open Source Project.

Android security overview. http://source.android.com/tech/ security/

Android. Security enhancements in android 4.2.

http://source.android.com/devices/tech/security/enhancements.html

Android.com. Android developers. http://www.android.com/

Barrera, D., Kayacik, H., van Oorschot, P.,and Somayaji, A.A methodology for empirical analysis of permission-based security models and its application to android. In Proc. of the ACM conference on Computer and Communications Security (2010).

Blasing, T., Batyuk, L., Schmidt, A., Camtepe, S., Albayrak, S.: An android application sandbox system for suspicious software detection. In: Malicious and Unwanted Software (MALWARE), International Conference on, IEEE (2010)

Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, ACM (2011)

Contagio mobile malware MiniDump.http://contagiominidump.blogspot.com/.

D. Barrera, H. G. Kayacik, P. C. van Oorschot, and A. Somayaji. A methodology for empirical analysis of permission-based security models and its application to android. In Proceedings of the ACM conference on Computer and communications security, ACM (2010).

Enck, W., Gilbert, P., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., and Sheth, A.N. TaintDroid: an Information-Flow Tracking System for Real-time Privacy Monitoring on Smartphones. Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI). October. Vancouver, BC, Canada: OSDI'10, Article No. 1-6. (2010).

Enck, W., Ongtang, M., and McDaniel, P. On Lightweight Mobile Phone Application. Proceedings of the 16th ACM conference on Computer and Communications Security. 9-13 NovemberChicago, IL, ACM (2009). Google play, 2012. https://play.google.com/store/apps.

Felt, A., Chin, E., Hanna, S., Song, D., and Wagner, D. Android Permissions Demystified. Proceedings of the 18th ACM conference on Computer and Communications Security(2011).

Google Inc. Android Developers. Manifest.permission. http://developer.android.com/reference/android/Manifest.permission.html, 2011.

Google Inc. Android Developers. Security and Permissions. http://developer.android.com/guide/topics/security/security.html, 2011.

J. Devesa, I. Santos, X. Cantero, Y. K. Penya, and P. G. Bringas, “Automatic Behaviour-based Analysis and Classification System for Malware Detection,” in Proceedings of the 12th International Conference on Enterprise Information Systems (ICEIS), (2010)

Johnson, R.,Wang, Z., Gagnon, C., Stavrou, A.: Analysis android applications permissions.In: Proceedings of the 6th International Conference on Software Security and Reliability. (2012)

K. Rieck, P. Trinius, C. Willems, and T. Holz, “Automatic Analysis of Malware Behavior using Machine Learning”,(2009).

L. Xie, X. Zhang, J. Seifert, and S. Zhu, "pBMDS: a behavior-based malware detection system for cellphone devices," in Proceedings of the third ACM conference on wireless network security (2010)

Yajin Zhou and Xuxian Jiang. Dissecting android malware: Characterization and evolution. In Security and Privacy (SP), 2012 IEEE