International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 24 Issue 5 – APRIL 2017.

K-NN BASED MALWARE DETECTION IN ANDROID SYSTEM

V.R.Niveditha[#1] and Dr. ShobaRani[*2]

[#] *M.TECH Information Security and Cyber Forensics, Dr. M.G.R. Educational and Research Institute University, Chennai, India*

[*] *Professor, Department of Computer Science and Engineering, Dr. M.G.R. Educational and Research Institute University, Chennai, India*

***Abstract*— In present life, the practices of human are being conquered by the mobile devices. It exhibits the functionalities similar to the personal computers. The emergences of android phones are everlastingly increases day by day. Thus, the attackers aim to steal the sensitive information from mobile devices. The proposed framework intends to develop a machine learning-based malware detection system on Android to detect malware applications and to develop security and privacy of smart phone users. This system monitors various API features and events obtained from the android applications, and analyses these characteristics by using machine learning classifiers to classify whether the application is benign or malware using k-nearest neighbouring algorithm. Experimental results have shown the effectiveness of the proposed model.**

***Index Terms*— Mobile devices, attackers, machine learning model, API features, k-nearest neighbour, security and privacy.**

I. INTRODUCTION

With the rapid growth in mobile computing technology, smart phones and tablets have evolved to offer stylish functionalities at lower costs. The Android platform has been in the front position of this mobile revolution and has gained enormous popularity over the last four years [1]. But the popularity has also brought the attention of major malware developers towards the platform. The fact that Android offers an open market model not like the closed app Store model of Apple, where each application (app) is manually inspected by security experts, makes it a more favourable target for malicious developers. The existence of many third-party app stores also contributes to the spreading of malicious apps for Android platform. The motivation for this research is to recognize the current state of malware research in Android smart devices, classify existing malware techniques and their countermeasures and through that process come up with novel suggestions for tackling current malwares. The main contributions of this paper are dual: (i) survey and classification of existing techniques and (ii) proposal of novel growth techniques and countermeasures.

There are different potential attack scenarios where an attacker can take advantage of the vulnerabilities of the Android platform to compromise a user. A possible scenario would be where a Trojan app performs some innocent job in the foreground, say download HD wallpapers, while it secretly leaks confidential private data such as contacts from users' mobile phone. In the case of the wallpapers app, it will contain INTERNET permission for downloading the wallpapers. An unsuspecting user might give not check the permissions requested and might allowance READ_CONTACTS permission as well accidentally. This data can be used for monetary benefits and/or propagating the malware by the attacker. In a different attack scenario, an attacker can try to kill the Smartphone of a victim by draining its battery life by excessive use of resource consuming services like radio, GPS etc [2]. These apps can be distributed as repackaged versions of popular apps such as ones which present location-based social media services. In this way, the user will be kept in the dark about private data leakage.

Each of the detection techniques can engage one of the three different approaches: static, dynamic or hybrid. The exact approach of an anomaly-based or signature-based method is determined by how information is gathered for use in malware detection. Anomaly based detection systems develop a prior training phase to establish a normality model for the system activity. In this method of detection, the detection system is first trained on the normal behavior of the application or target system to be monitored [3]. Using the normality model of behavior, it becomes possible to detect anomalous activities by looking for abnormal behavior or activities that deviate from the normal behavior earlier defined occurring in the system. Though this technique look more difficult, it has the advantage of being able to detect new and unknown malware attacks. Anomaly-based detection requires the use of feature vectors to train the classifier before successive classification can be carried out. These feature vectors are obtained from description or data collected from the system.

The rest of the paper is organized as follows: Section II describes the related work; Section III describes proposed work; Section IV represents experimental analysis and concludes in Section V.

II. RELATED WORK

This section depicts the variants malwares and the existing approaches in malware detection of android system.

A. *Different sorts of Malwares*

1) *Trojans:*

Trojans come into view to a user as a benign app [5]. In fact, they actually steal the user's secret information without the user's knowledge. Such apps can easily get right to use to the browsing history, mail, contacts and device IMEI numbers etc. of victim's device and steal this information without the approval of user. Fake Netflix [10] is an example of such malwares that provide user interface identical to original Netflix app and gather the user's login credentials. SMS Trojans exploit the premium services to bring upon yourself financial loss to the victim. Fake player is a well-known SMS Trojan that sends messages to best rate numbers without user awareness [11]. Z sone [12] and Android. Foney are also the examples of such SMS Trojan apps. Malwares also capture the user's banking information such as financial credit number and password. Zitmo and Spitmo Trojans are planned to steal the user's mTANs (Mobile Transaction Authentication Number) which then entire the transactions silently [13].

2) *Worms:*

Such malwares produce copies of it and distribute them over the network. For example, Bluetooth worms extend malware through the Bluetooth network by transfer copies of it to the paired devices. Android.Obad.OS is the illustration of Bluetooth worm [8].

3) *Spyware :*

Nick spy [11] and GPS Spy [14] are the example of spyware apps which appear as benign app, but it really monitors the user's secret information such as mail, contacts, bank mTANs, location etc. for some unwanted consequences. Personal spywares can install the malicious payload without the victim's information. It send the user's information such as text messages, contacts etc. to the attacker who install that software on victim's device [6].

4) *Botnets:*

Botnet is a network of compromise Android devices. Bot master, a remote server, control the botnet from side to side the C&C network. Geinimi [11] is one of the Android botnets.

5) *Ransom wares:*

Ransomware avoid the user from accessing their data on device by locking the device, until ransom sum is paid. FakeDefender.B [15] is a malware that masquerade itself as avast!, an antivirus. It locks the victim's device and power the user to pay ransom amount to release the device.

B. *Malware Detection in Android System*

In general, the android malwares are detected in two approaches, namely, static approaches and dynamic approaches [13].

1) *Static approach:*

Static approach verify the android functions without the applications usage. It is useful for finding malicious behaviours that may not operate until the particular condition occurs.

2) *Signature based approaches:*

The author in [14] suggested Andro Similar that detects the malwares used by repackaging and code obfuscation techniques. It is statistical signature process which is robust in nature. It generate the variable length signature for the application under test and compare it with the signatures in Andro Similar malware database and recognize the app as malware and benign on the basis of resemblance percentage. Authors tested the Andro Similar against 1260 apps among which 6779 apps were Google Play apps and 545 apps were from third party app store.

Droid Analytics is a signature based analytic system which take out and analyze the apps at op-code level. It not only generates the signature but also associate the malware with existing malwares after identifying the malicious content. It generates 3 level signatures. First it generates signature at process level by API call tracing then combining all the signatures of methods in a class it generates the class level signatures and at third level it generate the application signature by combine the signatures of the classes in the application.

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) *Permission based analysis:*

During installation, user must allow the app right to use all the resources requested by the app. Developers must mention the permissions requested for the resources in the AndroidManifest.xml file. But all confirmed permissions are not necessarily the required permissions for that specific application.

The author in [11] proposed a method for better detection of permission based malware detection which includes the analysis of both requested and required permissions as most of the time malware authors declare more permissions in the manifest file than they actually require for the application. Also it analyses the easy to retrieve features and then labels the application as benign or malware.

The author in [13] used a state machine based approach and formally analyzes the permission based Android security model. They also verified that the specified system satisfy the security property. The author in [14] proposed a Security Distance Model for mitigation of Android malware. Security Distance Model is based on the concept that not a single permission is enough for an application to threaten the security of Android devices. For example an application requesting permission READ_PHONE_STATE can access the phone number and IMEI but it cannot move data out of the device. There must be a combination of permissions to affect the security model of device such as INTERNET permission allows to concept the device with the network and will be needed to move data to some remote server.

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.

4) *Dalvik byte code analysis:*

The author in [7] developed SCANDAL, a static analyzer that analyze the Dalvik byte code of applications and detects the privacy leakage in applications. It determines the data flow from information source to any remote server. Dalvik byte code contains branch, method invocation and jump instructions which alters the order of execution of code and obfuscates the code. During execution, the possible paths that an application can take can be identified by the Byte code analysis. In [6] Authors have examined 90 applications from Android official market and 8 malicious applications from third party market place. They found privacy leakage in 11 Google market applications and 8 third party market applications. There is a need of performance optimization techniques to implement as SCANDAL consumes more time and memory for analysis of application. Also it does not support the applications which use reflections for data leakage. In the SCANDAL authors have implemented reflection semantics manually to detect the privacy leakage in malicious apps taken from black market.

Droid APIMiner [4], build upon Androgaurd [39], identifies the malware by tracking the sensitive API calls, dangerous parameters invoked and package level information within the byte code. To classify the application as benign or malware it implements KNN algorithm and detected up to 99 % accuracy and 2.2% false positive rate. The author in [5] presented SC android which analyze the Android application statically as they are installed and performs data flow analysis to checks whether the data flow through the applications is consistent or not. On the basis of data flows it declares the application as safe to be run with requested permissions. Authors use it as a security certification tool for Android apps.

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.

5) Dynamic approaches

Dynamic analysis examines the application during execution. It may neglect some of the code sections that are not executed but it can easily identify the malicious behaviours that are not detected by static analysis methods. Although static analysis methods are earlier to malware detection but they fail against the code obfuscation and encryption malwares.

6) Anomaly based detection

Crow Droid is used to detect the behavior of applications dynamically. Details of system calls invoked by the app are collected by the Strace tool and then crowd sourcing app, which is installed on the device, creates a log file and sends it to remote server. Log file may include the following information: Device information, apps installed on device and system calls. 2-mean clustering algorithm is applied at server side to classify the application as malware or benign. Results are stored at server database.

The author in [15] proposed Andromaly, a behavior based Android malware detection system. In order to classify the application as benign or malware it continuously monitor the different kind and patterns that indicate the device state such as battery level, CPU consumption etc. The machine learning algorithms is used to discriminate between malicious and benign apps. The solution can detect continuous attacks and can notify the user about these attacks.

7) Taint analysis

Taint Droid is the system-wide information flow tracking for Android. It can simultaneously track multiple sources of sensitive data such as camera, GPS and microphone etc. and identify the data leakage in third party developer apps. It labels the sensitive data and keeps track of that data and app when tainted data leaves moves from the device. It provides efficient tracking of sensitive information but it does not perform control flow tracking. Also, it cannot track information that leaves deice and returns in network reply.

8) Emulation based detection:

Android dynamic analysis platform Droid Scope, based on Virtual Machine Introspection was introduced. As the antimalware detect the presence of malwares because both of them reside in the same execution environment so the malwares also can detect the presence of antimalware. Droid Scope monitors the whole operating system by staying out of the execution environment and thus have more privileges than the malware programs.

Android Application Sandbox (AASandbox) which detects the suspicious applications by performing both static and dynamic analysis on them. It first extracts the .dex file into human readable form and then performs static analysis on application. Then it analyzes the low level interactions with system by execution of application in isolated sandbox environment. Actions of application are limited to sandbox due to security policy and do not affect the data on device. It uses Money tool to dynamically analyze the application behavior which randomly generates the user events like touches, clicks and gestures etc. It cannot detect the new malware types.

III. PROPOSED WORK

This section depicts general workflow of prediction and prevention of malware in android system. A devised machine learning systems is proposed to detect and preclude the malware apps. The API features are extracted, learned and stored in the android databases. The stored database contains the API features of malware apps and normal apps. This sensitive data path is studied for the prediction of malware behavior. An eminent k-nearest neighbour algorithm is used for differentiating the malicious and normal apps.

A simplified data mining approach is employed for detecting malware apps which consists of two phases, training and identification. The training phase trains the feature of malware and benign apps. Gently, the features of malware apps are trained to the android systems. Contrastingly, the identification phase identifies the unknown app whether it is malware or benign app. In order to detect the malware apps, the features and sensitive data transmission path are studied. These are studied from the API which consists of two features, a) the count of API calls from an app,b) count of API calls from the app components like activities, content providers task etc. The feature is extracted from the Dalvik byte code which is known as ‘Jimple’. It supports java source code and also Java byte code. The source codes in the android app contain significant and critical details to describe the behavior of an app.

The general facts of an android app are: i) most of the malware pose high level threats to the user's ii) 90% of the android phones control using SMS or networks iii) without user's knowledge, the sms or phone calls send to other users and iv) sensitive data is being stored on smart phones stealed. Hence, stealing and exploiting the sensitive data stands to be very outstanding features of the android malware. Along with these, machines learning approach is defined over the source and sink APIs. The source APIs associate with account, contact, SMS, database, and calendar. The sink APIs includes network, SMS, mails and files.

The features from the system calls are extracted and then associated to form call graph. It is further processed into three steps,

A. *Pre processing:*

In accord to Java, the android source code do not contain 'main method' and one or more components. By parsing androidmanifest.xml, the components and reachable call backs are obtained. Henceforth, the Jimple code can also be used for identifying the patterns of the receiver.

a) Building the call graph:

This method is simple and precise which scan the class definition code. Initially, the dummy main method is generated which contain class definition code. All the collected call backs are registered with dummy main method. Thus, the global graph is formed. If the broadcast receiver is instantiated with register receiver call, then we can determine that receiver component is dynamically registered.

B. *Connect asynchronous calls:*

A handler variable is used for handling the call back method which processes the messages via methods like send Message and sendEmptyMessage.

Based on the above pre processed outcomes, the classifier model is constructed. The target of the classifier model is to distinguish the malware and benign apps. It is processed in two steps. The first step includes the LR algorithm that calculates the weight values for malicious apps. Based on obtained weight values, top k weight score is measured. Thus, APIs list are trained and used for future process. Mahalanob is distance is measured between two features of unknown apps.

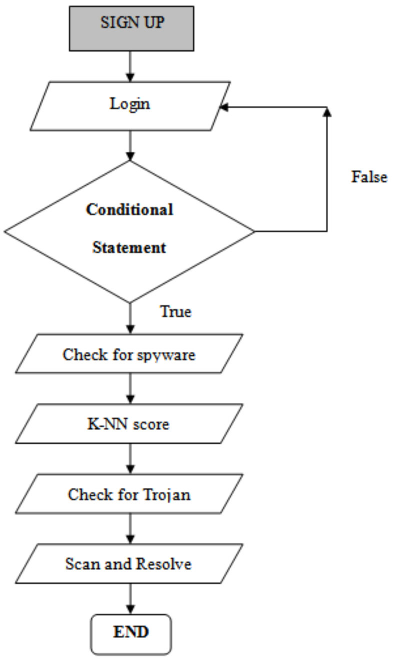


Fig.1. Proposed block diagram

IV. EXPERIMENTAL ANALYSIS

This section depicts the experimental analysis carried out to justify the proposed model.

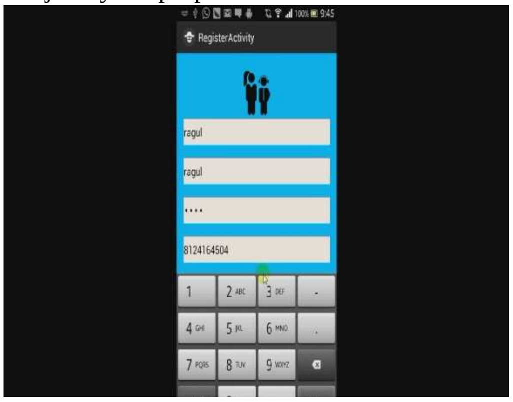


Fig.2. Enrolment of android users

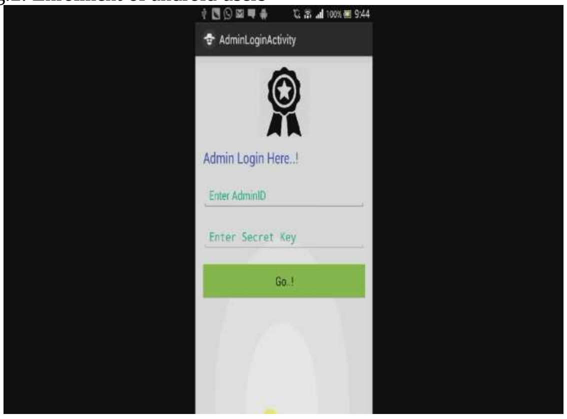


Fig.3. Login page of admin

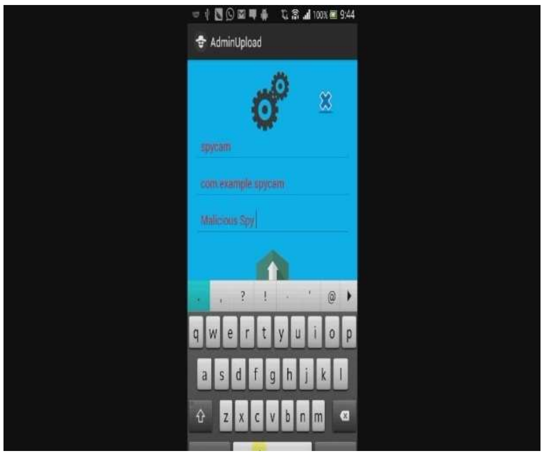


Fig.4 File uploads by admin

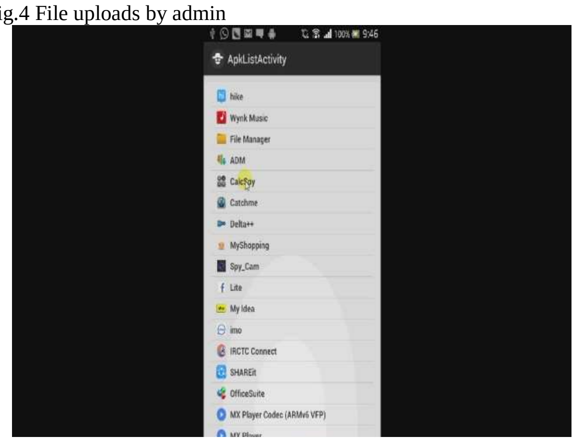
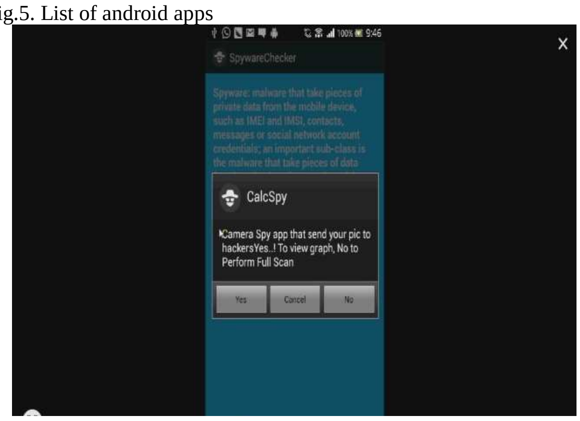


Fig.5. List of android apps



Spyware: malware that take pieces of private data from the mobile device, such as IMEI and IMSI, contacts, messages or social network account credentials; an important sub-class is the malware that take pieces of data

CalcSpy

Camera Spy app that send your pic to hackersYes..! To view graph, No to Perform Full Scan

Fig.6. Detection of spywares

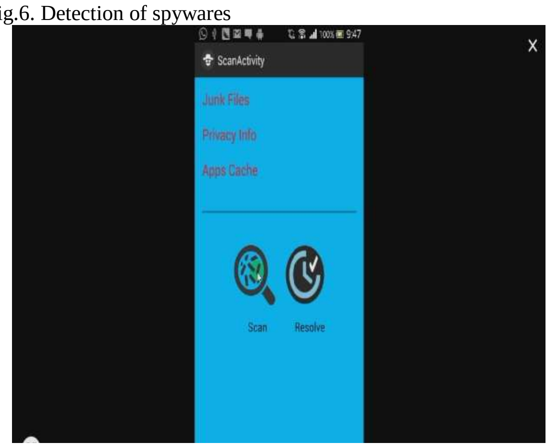


Fig.7. Scanning the android apps to check the spam data

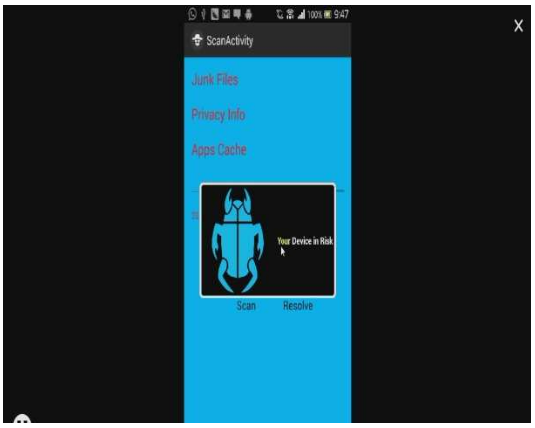


Fig.8. Displaying risk level of the Android phone

V. CONCLUSION

Android OS is the most adopted operating systems by the smart phone users. Due to its popularity, different applications and adware's are introduced increasingly. A different commercial signature tools are available in the market to prevent incursion and distribution of the malicious applications. Numerous researches have been conducted which claims that traditional signature based detection system work well up to confident level and malware authors use numerous techniques to evade these tools. In this paper, we have proposed an efficient k-Nearest Neighbour (k-NN). The target of the study is to distinguish the malware apps and normal apps. Firstly, the API features are trained into android databases. Then the unknown android app is identified with the help of trained API features. Eventually, the weight score between two features are used to distinguish the malware apps from benign apps. Experimental results have shown the effectiveness of the proposed algorithm.

REFERENCES

[1] Andrea Saracino, Daniele Sgandurra, Gianluca Dini and Fabio Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention", IEEE Transactions on Dependable and Secure Computing, 2016.

[2] Y. Zhou, X. Jiang, Dissecting android malware: Characterization and evolution, in: Security and Privacy (SP), 2012 IEEE Symposium on, 2012, pp.95–109.

[3] L. Li, A. Bartel, T. F. Bissyand´e, J. Klein, Y. Le Traon, S. Arzt, S. Rasthofer, E. Bodden, D. Octeau, P. McDaniel, Iccta: Detecting inter-component privacy leaks in android apps, in: Proceedings of the 37th Inter-national Conference on Software Engineering - Volume 1, ICSE '15, IEEE Press, Piscataway, NJ, USA, 2015, pp. 280–291.

[4] A. P. Felt, M. Finifter, E. Chin, S. Hanna, D. Wagner, A survey of mobile malware in the wild, in: Proceedings of the 1st ACM Workshop on Security and Privacy in Smart phones and Mobile Devices, SPSM '11, ACM, New York, USA, 2011, pp. 3–14.

[5] A. P. Felt, E. Chin, S. Hanna, D. Song, D. Wagner, Android permissions demystified, in: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS '11, ACM, New York, USA, 2011, pp. 627–638.

[6] P. Hornyack, S. Han, J. Jung, S. Schechter, D. Wetherall, These aren't the droids you're looking for: Retrofitting android to protect data from imperious applications, in: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS '11, ACM, New York, USA, 2011, pp. 639–652.

[7] Z. Yang, M. Yang, Y. Zhang, G. Gu, P. Ning, X. S. Wang, Appintent: Analyzing sensitive data transmission in android for privacy leakage detec-tion, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS '13, ACM, New York, USA, 2013, pp.1043–1054.

[8] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, P.P. McDaniel, Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps, SIGPLAN Not. 49 (6) (2014) 259–269.

[9] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, “Android permissions: user attention, comprehension, and behavior,” in Symposium On Usable Privacy and Security, SOUPS ’12, Washington, DC, USA - July 11 - 13, 2012, 2012, p. 3.

[10] Y. Zhou and X. Jiang, “Dissecting android malware: Characterization and evolution,” in Proceedings of the 2012 IEEE Symposium on Security and Privacy, ser. SP ’12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 95–109.

[11] Y. Zhou and X. Jiang, “Dissecting android malware: Characterization and evolution,” in Proceedings of the 2012 IEEE Symposium on Security and Privacy, ser. SP ’12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 95109.[Online].Available:http://dx.doi.org/10.1109/SP.2012.16

[12] “Contagio mobile, mobile malware mini dump.” [Online]. Available: http://contagiominidump.blogspot.com

[13] Google Groups, “Virustotal,” 2015. [Online]. Available: https: //www.virustotal.com/

[14] Dr.Web, “Android malware review,” 2015. [Online]. Available: http://news.drweb.com/show/review/?lng=en&i=9546

[15] K. S. Labs, “Kindsight security labs malware report h1 2014,” 2014. [Online]. Available: http://resources.alcatel-lucent.com/ ?cid=180437.