Enhanced Multilevel Anomaly Detection for Android Malware

Dr. Santhi Baskaran[1], G.Maheshwari[2], J. Pearly percy[3], P.Priyadharshini[4]

[1]*Professor, Department of Information Technology, Pondicherry Engineering College, Pillaichavady, Puducherry, India*
[2]*Student, Department of Information Technology, Pondicherry Engineering College, Pillaichavady, Puducherry, India*
[3]*Student, Department of Information Technology, Pondicherry Engineering College, Pillaichavady, Puducherry, India*
[4]*Student, Department of Information Technology, Pondicherry Engineering College, Pillaichavady, Puducherry, India*

Abstract — *Android device users are frequently threatened by an increasing number of malicious applications, generally called malware. Malware constitutes a serious threat to user privacy, money, devices and file integrity. We can classify malware into small number of behaviours and classes, each of which performs a limited set of misbehavior that characterize them. This misbehavior can be defined by monitoring features belonging different android levels. In this project, we present Enhanced Multilevel Anomaly Detection for Android Malware (EMADAM), a novel host based malware detection system for android devices which simultaneously analyses and correlates features at four levels: kernel, application, user and package, detect and stop malicious behaviors.*

Index terms — *Malware detection, Multilevel anomaly, Malicious actions, Android behavioural patterns*

I. INTRODUCTION

Smartphones and tablets have become extremely popular in the last years. At the end of 2014, the number of active mobile devices worldwide was almost 7 billions, and in developed nations the ratio between mobile devices and people is estimated as 120.8% [1]. Given their large distribution, and also their capabilities, in the last two years mobile devices have became the main target for attackers [2]. Android, the open source operative system (OS) introduced by Google, has currently the largest market share [1], which is greater than 80%. Due to the openness and popularity, Android is the main target of attacks against mobile devices (98.5%), with more than 1 million of malicious apps currently available in the wild [3]. Malicious apps (generically called malware) constitute the main vector for security attacks against Mobile Devices. The malwares in user device threatens the user privacy, the device integrity, or even user's credit. Some common examples of attacks performed by Android malicious apps are stealing contacts, login credentials, text messages, or maliciously subscribing the user to costly premium services. Furthermore, all these misbehaviors can be performed on Android devices without the user noticing them.

It has been recently reported that almost 60% of existing malware send stealthy premium rate SMS messages. However, also Google Play, the official market for Android apps, has hosted apps which have been found to be malicious. Along with the vast increase of Android malware, several security solutions have been proposed by the research community, spanning from static or dynamic analysis of apps [4], to applying security policies enforcing data security [5] [6], to run-time enforcement [7] [8]. However, these solutions still present significant drawbacks. In particular, they are attack-specific, i.e. they usually focus on and tackle a single kind of security attack, e.g. privacy leaking [7] [8], or privilege escalation (jail-breaking) [5] [9]. Moreover, these frameworks generally require a custom OS [8].

In particular, to detect app misbehaviors, EMADAM monitors the device actions, its interaction with the user and the running apps, by retrieving five groups of features at four different levels of abstraction, namely the kernel level, application-level, user-level and package-level. For some groups of features EMADAM applies an anomaly based approach, for other groups it implements a signature based approach that considers behavioral patterns that we have derived from known malware misbehaviors. In fact, EMADAM has been designed to detect malicious behavioral patterns extracted from several categories of malware. This multi-level behavioral analysis allows EMADAM to detect misbehaviors typical of almost all malware which can be found in the wild. EMADAM also has shown efficient detection capabilities as it introduces an 1.4% performance overhead and a 4% battery depletion.

Finally, EMADAM is usable because it both requires little-to-none user interaction and does not impact the user experience due to its efficiency. EMADAM achieves the above goals as follows: (i) it monitors five groups of Android features, among which system calls (type and amount) globally issued on the device, the security relevant API calls, and the user activity, to detect unusual user and device behavioral patterns; to this end, it exploits two cooperating proximity-based classifiers to detect and alert anomalies; (ii) it intercepts and blocks dangerous actions by detecting specific behavioral patterns which take into account a set of known security hazard for the user and the device; (iii) every time a new app is installed, EMADAM assesses its security risk by analyzing the requested permissions and reputation metadata, such as user scores and download number, and it inserts the app in a suspicious list if evaluated as risky.

II. LITERATURE SURVEY

Researchers in [13] find that Cloaker, a stealthy rootkit, exploits features of the ARM processor in order to hide itself. There are two specific ARM hardware features utilized by Cloaker, the ability to change the location of the interrupt vector and the ability to lock addresses in the translation look aside buffer. This second technique allows memory to be stealthily mapped into processes without modifying the OS level (detectable) page table entries.

This work presents AppGuard[8], a powerful and flexible security system that overcomes these deficiencies. It enforces user-defined security policies on untrusted Android applications without requiring any changes to a Smartphone's firmware, root access, or the like. Fine-grained and state full security policies are expressed in a formal specification language, which also supports secrecy requirements.

Our system offers complete mediation of security-relevant methods based on calleesite inline reference monitoring and supports widespread deployment. In the experimental analysis we demonstrate the removal of permissions for overly curious apps as well as how to defend against several recent real-world attacks on Android phones. Our technique exhibits very little space and runtime overhead. The utility of AppGuard has already been demonstrated by more than 1,000,000 downloads.

TaintDroid incurs only 14% performance overhead on a CPU-bound micro-benchmark and imposes negligible overhead on interactive third-party applications. Using TaintDroid to monitor the behavior of 30 popular third-party Android applications, we found 68 instances of potential misuse of users' private information across 20 applications. Monitoring sensitive data with TaintDroid provides informed use of third-party applications for phone users and valuable input for Smartphone security service firms seeking to identify misbehaving applications.

In this paper CopperDroid[1], an approach built on top of QEMU to automatically perform out-of-the-box dynamic behavioural analysis of Android malware. To this end, CopperDroid presents a unified analysis to characterize low-level OS-specific and high-level Android-specific behaviors. Based on the observation that such behaviors are however achieved through the invocation of system calls, CopperDroid's VM-based dynamic system call-centric analysis is able to faithfully describe the behavior of Android malware whether it is initiated from Java, JNI or native code execution.

We carried out extensive experiments to assess the effectiveness of our analyses on a large Android malware data set of more than 1,200 samples belonging to 49 Android malware families (provided by the Android Malware Genome Project) and about 400 samples over 13 families (collected from the Contagio project). Our experiments show that a proper malware stimulation strategy (e.g., sending SMS, placing calls) successfully discloses additional behaviors on a non-negligible portion of the analyzed malware samples.

Android's security framework has been an appealing subject of research in the last few years. Android has been shown to be vulnerable to application-level privilege escalation attacks, such as confused deputy attacks, and more recently, attacks by colluding applications. While most of the proposed approaches aim at solving confused deputy attacks, there is still no solution that simultaneously addresses collusion attacks.

Android's permission system is intended to inform users about the risks of installing applications. When a user installs an application, he or she has the opportunity to review the application's permission requests and cancel the installation if the permissions are excessive or objectionable. We examine whether the Android permission system is effective at warning users. In particular, we evaluate whether Android users pay attention to, understand, and act on permission information during installation. We performed two usability studies: an Internet survey of 308 Android users, and a laboratory study wherein we interviewed and observed 25 Android users. Study participants displayed low attention and comprehension rates: both the Internet survey and laboratory study found that 17% of participants paid attention to

permissions during installation, and only 3% of Internet survey respondents could correctly answer all three permission comprehension questions.

This indicates that current Android permission warnings do not help most users make correct security decisions. However, a notable minority of users demonstrated both awareness of permission warnings and reasonable rates of comprehension. We present recommendations [11] for improving user attention and comprehension, as well as identify open challenges.

III. PROBLEM DEFINITION

This study involves the malware detection process for the Android platform. There are known and unknown malware and benign apps in the market. Known malware is removed from the market place. Unknown malware evades the detection engine by hiding its malicious activities. We identify these unknown malwares and the benign apps based on binary classification. If there are a finite set of classes, then the classifier will determine the class of a given object. Binary classification involves only two sets of possible classes. Most of the discussed systems are attack specific, usually focus on and tackle a single kind of security attack (e.g,) Privacy leaking or privilege escalation. Most of the existing works detects malware either in static mode or dynamic mode but not both. Moreover, these frameworks generally require a custom OS. Apart from this ad-hoc security solutions, this work acts as an attempt to limit set of dangerous operations that external applications can perform.

IV. PROPOSED SYSTEM

In proposed system, we present a novel multi-level and behavior based, malware detector for Android devices called EMADAM. In particular, to detect app misbehaviors, EMADAM monitors the device actions, its interaction with the user and the running apps, by retrieving five groups of features at four different levels of abstraction, namely the kernel level, application-level, user-level and package-level. For some groups of features EMADAM applies an anomaly based approach, for other groups it implements a signature based approach that considers behavioral patterns that we have derived from known malware misbehaviors. In fact, EMADAM has been designed to detect malicious behavioral patterns extracted from several categories of malware. This multi-level behavioral analysis allows EMADAM to detect misbehaviors typical of almost all malware which can be found in the wild. Finally, EMADAM is usable because it both requires little-to-none user interaction and does not impact the user experience due to its efficiency. Fig. 1 explains the static detection of malicious contents using the apk file of the application.

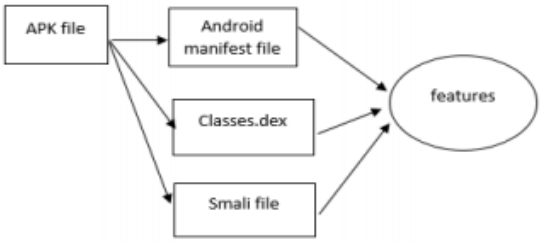

Fig. 1 Feature extraction from apk file

The main novelty of EMADAM is its cross-layer approach and a novel integration of techniques (some of which already existing) that provides high efficacy with low overhead. EMADAM has been conceived to prove that a multilevel approach makes it possible to dynamically detect most of current Android malware, right on the device with limited overhead. To verify that such approach is indeed viable, a large extensive set of tests have been performed to prove empirically its efficacy.

V. ARCHITECTURE OF EMADAM

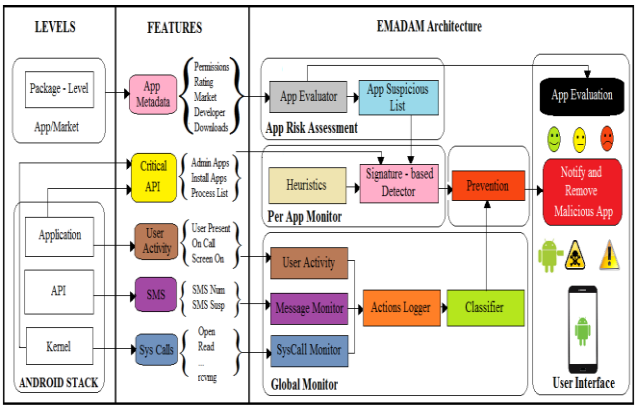

Figure2. Architectural diagram of EMADAM

VI. LEVELS OF ANALYSIS AND FEATURES

Level	Group	Feature	Description	Targeted Misbehavior
Kernel	Sys Calls	open, read, . . .	System calls concerning file and inter-component communication	Sudden and unmotivated activity increase

Application	SMS	Number of SMS (SMS Num)	Amount and recipient of outgoing SMS	Unsolicited outgoing messages
Application	SMS	Suspicious SMS (SMS Susp)	Amount of SMS sent to recipients not in contacts	Spyware or registration to premium services
Application	Critical API	Administrator App	Verify if an app attempts to get admin privileges	Apps which attempt to take control of the device
Application	Critical API	New App Installation	Verify if an app attempts to install a new one	Unauthorized app installations
Application	Critical API	Process List	Verify if an app generates high number of processes	Buffer overflow (Rootkit) attacks
Application	Critical API	Critical SysCalls	Amount of critical system calls generated by an app	Apps that access files and resources in backround (Spyware, Botnet and Trojan)
User	User Activity	On Call	Verify if a phone call is ongoing	Unsolicited activities of Spyware, Botnet, Installer and Rootkit
User	User Activity	Screen On	Verify if the device screen is on	Unsolicited activities of Spyware, Botnet, Installer and Rootkit
Package	App Metadata	Permissions requested (manifest.xml)	Riskiness of app	Suspicious requests of dangerous permissions
Package	App Metadata	Market Info (User scores, . . .)	Popularity of app	Trojan

Table 1: Levels of analysis and features at different levels

VII. MODULES

EMADAM can be logically decomposed into four main architectural blocks: App Risk Assessment, Global Monitor, Per App Monitor**,** User Interface & Prevention and Deployment**.** The first one is the App Risk Assessment, which includes the App Evaluator that implements an analysis of metadata of an app package (apk) (permission and market data), before the app is installed on the device. This evaluation computes the app's risk score, i.e. the likelihood that the app is a malware. Based on this risk evaluation, this component populates a set of suspicious apps (App Suspicious List), which will be monitored at run-time. The second block is the Global Monitor, which monitors the device and OS features at three levels, i.e. kernel (SysCall Monitor), user (User Activity Monitor) and application (Message Monitor). These features are monitored regardless of the specific app or system components generating them, and are used to shape the current behavior of the device itself. Then, these behaviors are classified as genuine (normal) or malicious (anomalous) by the Classifier component. The third block is the Per-App Monitor, which implements a set of known behavioral patterns to monitor the actions performed by the set of suspicious apps (App Suspicious List), generated by the App Risk Assessment, through the Signature-Based Detector. Finally, the User Interface & Prevention component includes the Prevention module, which stops malicious actions and, in case a malware is found, handles the procedure for removing malicious apps using the User Interface (UI). The UI handles notifications to device user, in particular:

- **(i)** The evaluation of the risk score of newly-downloaded apps by the App Evaluation.
- **(ii)** The reporting of malicious app (Notify) and
- **(iii)** To ask the user whether to remove them (Remove Malicious App).

A. APP RISK ASSESSMENT

When a new app is installed on the device (deploy-time), the App Evaluator component intercepts and hijacks the installation event. This component analyzes the metadata of the new app to assess its risk, by retrieving features from the app package, related to critical operations, and from the market, related to app reputation. In detail, these features are: (i) the permissions declared in the manifest, (ii) the market of provenance, (iii) the total number of downloads, (iv)the developer reputation and (v) the

user rating. The five parameters are analyzed through a hierarchical algorithm which returns a decision on the riskiness of the app classifying it as safe or risky. Based on this decision, the user can choose whether to continue the installation (or not) of the new app. If the user chooses to install a risky app, its package name is recorded in the EMADAM App Suspicious List and is continuously monitored looking for the known behavioral patterns. Note that EMADAM extracts all these pieces of information in a process which is totally transparent to the user.

The user can, however, decide whether she prefers to receive a notification of the decision of the App Evaluator, or to keep the process invisible. In the following, we assume that the user chooses the transparent approach (i.e., new apps are always installed, but inserted into the App Suspicious List if risky), as to allow EMADAM to enforce security policies on the device. It is worth noting that the App Evaluator is not a detector of malicious apps. Instead, the App Evaluator aims at finding apps which are risky, which should be monitored at run-time by EMADAM, improving the overall performance.

B. GLOBAL MONITOR

The Global Monitor is at the core of the EMADAM framework, since it is responsible of collecting the run-time device behaviors and classifying them as "genuine" or "malicious". In EMADAM, a behavior is represented through a vector of features. For each of them, EMADAM records how many times a specific feature has been used in a period of time Tk. The features are extracted from different kinds of dynamic events : User Activity, Critical API (in particular, SMS, i.e. text messages) and System Call (Sys Calls). The Actions Logger is the component that records all these features into a vector, which is then fed to the Classifier. This component is trained to recognize genuine behaviors related to normal device usage, and malicious behavioral patterns deviating from the genuine ones, derived from the seven classes of malware. The classifier correlates features from the three monitored levels, and detects misbehaviors which could pass unnoticed if monitored separately on the single levels. The Global Monitor is effective in detecting malicious behaviors, especially for SMS Trojan, Rootkit, Installers and Ransomware.

1) User Activity and Message Monitor: The User Activity and Message Monitor allow EMADAM to intercept calls to security relevant API functions, namely related to SMS messages and user activity. As we have previously recalled, these features are critical from a security point of view to detect SMS sent to premium-numbers and/or without the user knowledge. EMADAM hijacks security relevant methods, by monitoring their actual parameters and controlling the final outcome of the action. In particular, EMADAM hijacks the SendTextMessage() and SendDataMessage() methods to control the events of outgoing SMS messages . Furthermore, using standard Android APIs, EMADAM also verifies

(i) if the user is interacting with the device,

(ii) if the device screen is on/off and

(iii) if a phone call is ongoing.

These elements are used to assess when the user is active. In particular, we can categorize the status of the user as being in one of two possible states (active or idle), which are strongly dependent on the activity of the phone itself. In the first user activity state (active) either (i) is on, or (ii) the screen is off but a phone call is ongoing. In fact, when the user is active, the phone has to show interactive contents on the screen and receives inputs from the user, or handles the elements involved in a phone call. Otherwise, in the second user activity state (idle), the phone is not active.

2) Action Logger and Classifier: Among all the features that EMADAM collects, the Action Logger retrieves 14 features from three classes at three distinct levels (kernel, application, user). In detail, the first eleven features concern the system calls related to file modification and inter-component communication (i.e., open, ioctl, brk, read, write, exit, close, sendto, sendmsg, recvfrom, and recvmsg).

C. PER-APP MONITOR

The Per-App Monitor component is complementary to the Global Monitor since it is aimed at detecting additional, signature-based, known misbehaviors. The Per-App monitor is based on a set of known malicious behavioral patterns which considers the Suspicious App List created by the App Risk Assessment module, the alerts raised by the classifiers and a set of features at application-level not considered by the classifier. The Per-App monitor exploits behavioral patterns which represent suspicious behaviors that have been inferred by analyzing the behavioral classes of malware at API level and kernel level.

To consider these behavioral patterns, Per-App Monitor constantly monitors three features, namely:

(i) The list of apps with administrator privileges, which are those apps that can access a specific set of dangerous security

relevant API and that cannot be removed unless the privileges are revoked,

(ii) The SMS default app, which is the app that by default handles the operations related to text messages and that can be changed by the user,

(iii) The app in foreground, which is the app currently interacting with the user.

D. USER INTERFACE & PREVENTION

The User Interface & Prevention includes the Prevention module that acts as a security enforcement mechanism by blocking the detected misbehaviors related to behavioral patterns, e.g. a SMS being sent without the user authorization. In such a case, the User Interface (UI) module handles the process for removing the responsible app.

The UI conveys to the user all the events which require an active interaction, such as for removing malicious apps, and is also used by the user to select which behaviors should be blocked or allowed. Finally, the UI is exploited by the App Evaluator to communicate to the user the risk score of a new app at deploy-time. In this case, the user can then decide whether to continue the installation (or not) of the app.

E. DEPLOYMENT

EMADAM comes as a package which contains the EMADAM apk7, implementing the User Interface and Prevention Modules, the App Evaluator, the Per-App Monitor, and Global Monitor. The EMADAM package also contains the Superuser8 apk, for handling attempts of accessing root privileges, and the X-posed Installer apk, for hooking and handling events relevant to the Per-App Monitor. When installed, EMADAM deploys a kernel monitoring module, by issuing the insmod command. This command, and the X-posed Installer, requires root access.

VIII. RELATED WORKS

Taintdroid[5] is a security framework for android devices which tracks information flow to avoid malicious stealing of sensitive information. Differently from EMADAM TaintDroid targets a very specific class of attacks. Moreover, TaintDroid requires a custom ROM of the Android system, to implement the information flow mechanisms. A behavioral analysis of Android apps at the system call level is presented in [35]. The authors propose a framework called CopperDroid that discerns good behaviors from bad ones, by automatically stimulating malicious apps to misbehave through instrumentation. The analysis of behaviors is automatic, which means that the behavior of the stimulated app by user interaction is not considered as in EMADAM. Android Security Framework (ASF) [34] is a generic and extensible security framework for Android that provides security API to facilitate the inclusion of security extensions in Android. This approach is orthogonal to EMADAM: the goal of EMADAM is to detect malware, i.e. anomalies, while ASF is more oriented to the enforcement of policies. The authors of [31] presents a system which aim at detecting root kit hidden in Trojanized apps. This framework, Droid Analyzer, identifies the features which are typical of root kits and then looks for them statically in the code of apps, performing the analysis on an external server. On the contrary, EMADAM performs the analysis on the mobile device, and is focused on several classes of malware. MOSES [36] is a policy-based framework that enforces software isolation of apps (and data) on Android. EMADAM is more focused on malware detection, even if it allows users to define some high-level policies for apps.

Alterdroid [33] is a tool that compares the differences in behavior between an original app and automatically generated version that contain modifications (faults) to detect hidden malware, such as in pictures. Differently from EMADAM, Alterdroid performs static analysis and does not target general malware, being not able to detect pieces of malware that do not hide malicious code in static resources. [37] proposes a method for malware detection based on embeddings of function call graphs in a vector space capturing structural relationship. This representation is used to detect Android malware using machine learning techniques by achieving a good accuracy. Similarly, [32] classifies Android malware via dependency graphs by extracting a weighted contextual API dependency graph as program semantics to construct feature sets. These approaches implement a static detection of Android malware while EMADAM implements both static and runtime analysis. [38] statically analyzes app to derive a set of features for malware detection at application-level and evaluates several classifiers for Android apps. EMADAM also analyzes system calls and user activities and classifies the activities at run-time. Similarly, DREBIN [39] performs static analysis of Android apps to gather features that are embedded in a joint vector space, such that typical patterns indicative for malware can be automatically identified and used for explaining the decision. [40] presents a method for screening malicious Android apps that uses the requested permissions and a metric that measures the riskiness of an app based on a data-flow graph. These data are used with a set of machine learning algorithms to classify new apps as malicious or benign with an accuracy of 96% with less than 1% false positives. An approach similar to the App Classifier of EMADAM is presented in [41], which

proposes to communicate an index assessing the risk level of an Android application. However, the proposed index is mainly intended for a comparison between similar apps, pushing the user to choose the less risky.

IX. CONCLUSION

Starting from the end of 2011, attackers have increased their efforts toward Android Smartphones and tablets, producing and distributing hundreds of thousands of malicious apps. These apps threaten the user data privacy, money and device integrity, and are difficult to detect since they apparently behave as genuine apps bringing no harm. This project proposes EMADAM, a multi-level host-based malware detector for Android devices. By analyzing and correlating several features at four different Android levels, EMADAM is able to detect misbehaviors from malware behavioral classes that consider 125 existing malware families, which encompass most of the known malware. To the best of our knowledge, EMADAM is the first system which aims at detecting and stopping at run-time any kind of malware, without focusing on a specific security threat, using a behavior-based and multi-level approach.

REFERENCES

[1] "Global mobile statistics 2014 part a: Mobile subscribers; handset market share; mobile operators," http://mobiforge.com/ research-analysis/global-mobile-statistics-2014-part-a-mobilesubscribers-handset-market-share-mobile-operators, 2014.

[2] "Sophos mobile security threat reports," 2014, last Accessed: 20 November 2014. [Online].Available: http://www.sophos.com/ en-us/threat-center/mobile-security-threat-report.aspx

[3] M. G. Christian Funk, "Kaspersky security bullettin 2013," December 2013. [Online]. Available: http://media.kaspersky. com/pdf/KSB 2013 EN.pdf

[4] A. Reina, A. Fattori, and L. Cavallaro, "A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors," EuroSec, April, 2013.

[5] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A. Sadeghi, and B. Shastry, "Towards taming privilege-escalation attacks on android," in 19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012, 2012.

[6] M. Backes, S. Gerling, C. Hammer, M. Maffei, and P. von StypRekowsky, "Appguard fine-grained policy enforcement for untrusted android applications," in Data Privacy Management and Autonomous Spontaneous Security, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 213–231.

[7] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, "Taming information-stealing smartphone applications (on android)," in Proceedings of the 4th International Conference on Trust and Trustworthy Computing, ser. TRUST'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 93–107. [Online]. Available: http: //dl.acm.org/citation.cfm?id=2022245.2022255

[8] [8] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," in Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924971

[9] [9] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, and B. Shastry, "Practical and lightweight domain isolation on android," in Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, ser. SPSM '11. New York, NY, USA: ACM, 2011, pp. 51–62. [Online]. Available: http://doi.acm.org/10.1145/2046614.2046624

[10] [10] 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.

[11] [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. 95–109. [Online]. Available: http://dx.doi.org/10.1109/SP.2012.16

[12] [12] Schlegel, R. et al., 2011. Soundcomber: A stealthy and context-aware sound trojan for smartphones. Proceedings of the. Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Sou ndcomber+:+A+Stealthy+and+ContextAware+Sound+Trojan+for+Smartphones#0.

[13] [13] David, F. & Chan, E., 2008.Cloaker: Hardware supported rootkit concealment.Security and Privacy. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4531160 [Accessed February 2, 2014].

[14] F. Del Bene, G. Dini, F. Martinelli, I. Matteucci, M. Petrocchi, A. Saracino, and S. D., "Risk analysis of android applications: A multi-criteria and usable approach," Consiglio Nazionale delle Ricerca - Istituto di Informatica e Telematica, Tech. Rep. TR-04- 2015, 2015. [Online]. Available: http://www.iit.cnr.it/node/32795

[15] C. Gates, J. Chen, N. Li, and R. Proctor, "Effective risk communication for android apps," Dependable and Secure Computing, IEEE Transactions on, vol. 11, no. 3, pp. 252–265, May 2014.

[16] G. Dini, F. Martinelli, A. Saracino, and D. Sgandurra, "Madam: A multi-level anomaly detector for android malware," in Computer Network Security, ser. Lecture Notes in Computer Science, I. Kotenko and V. Skormin, Eds. Springer Berlin Heidelberg, 2012, vol. 7531, pp. 240–253.

[17] T. C., "Say goodbye to custom stock roms and hello to xposed framework," May 2013. [Online]. Available: http://www.xda-developers.com/android/say-goodbyeto-custom-stock-roms-and-hello-to-xposed-framework/

[18] D.-K. Kang, D. Fuller, and V. Honavar, "Learning classifiers for misuse and anomaly detection using a bag of system calls representation," in Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC, June 2005, pp. 118–125.

[19] D. Mutz, F. Valeur, G. Vigna, "Anomalous System Call Detection," ACM Transactions on Information and System Security, vol. 9, no. 1, pp. 61–93, February 2006.

[20] G. Vigna, W. Robertson, and D. Balzarotti, "Testing networkbased intrusion detection signatures using mutant exploits," in Proceedings of the 11th ACM

Conference on Computer and Communications Security, ser. CCS '04. New York, NY, USA: ACM, 2004, pp. 21–30. [Online]. Available: http://doi.acm.org/10.1145/1030083.1030088

[21] T. M. Cover, P.E. Hart, "Nearest Neighbor Pattern Classification," IEEE Transactions on Information Theory, vol. IT-13, no. 1, pp. 21–27, January 1967.

[22] O. Kramer, "Dimensionality reduction by unsupervised k-nearest neighbor regression," in Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on, vol. 1, Dec 2011, pp. 275–278.

[23] A. Developer, "Android smsmanager api reference page," 2015. [Online]. Available: http://developer.android.com/reference/android/telephony/SmsManager.html

[24] V. Misra, "What are the exact mechanisms/flaws exploited by the "rage against the cage" and "z4root" android exploits?"[Online].Available:http://www.quora.com/What-are-the-exactmechanisms-flaws-exploited-by-the-rage-against-the-cage-andz4root-Android-exploits

[25] B. Wolfe, K. Elish, and D. Yao, "Comprehensive behavior profiling for proactive android malware detection," in Information Security, ser. Lecture Notes in Computer Science, S. Chow, J. Camenisch, L. Hui, and S. Yiu, Eds. Springer International Publishing, 2014, vol. 8783, pp. 328–344. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-13257-0 19

[26] H. Kayacik and A. Zincir-Heywood, "Mimicry attacks demystified: What can attackers do to evade detection?" in Privacy, Security and Trust, 2008. PST '08. Sixth Annual Conference on, Oct 2008, pp. 213–223.

[27] M. J. Darnell, "Acceptable system response times for tv and dvr," in Proceedings of the 5th European Conference on Interactive TV: A Shared Experience, ser. EuroITV'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 47–56. [Online]. Available: http://dl.acm.org/citation.cfm?id=1763017.1763025

[28] "How antivirus affect battery life," https://www.luculentsystems. com/techblog/minimize-battery-drain-by-antivirus-software/, last accessed on 23/02/2015.

[29] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: An information flow tracking system for real-time privacy monitoring on smartphones," Commun. ACM, vol. 57, no. 3, pp. 99–106, Mar. 2014. [Online]. Available: http://doi.acm.org/10.1145/2494522

[30] M. Sun, M. Zheng, J. C. S. Lui, and X. Jiang, "Design and implementation of an android host-based intrusion prevention system," in Proceedings of the 30th Annual Computer Security Applications Conference, ser. ACSAC '14. New York, NY, USA: ACM, 2014, pp. 226–235. [Online].Available:http://doi.acm.org/10.1145/2664243.2664245

[31] S.-H. Seo, A. Gupta, A. M. Sallam, E. Bertino, and K. Yim, "Detecting mobile malware threats to homeland security through static analysis," Journal of Network and Computer Applications, vol. 38, no. 0, pp. 43 – 53, 2014.

[32] M. Zhang, Y. Duan, H. Yin, and Z. Zhao, "Semantics-aware android malware classification using weighted contextual api dependency graphs," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 1105–1116. [Online]. Available:http://doi.acm.org/10.1145/2660267.2660359

[33] G. Suarez-Tangil, J. Tapiador, F. Lombardi, and R. Di Pietro, "Thwarting obfuscated malware via differential fault analysis," Computer, vol. 47, no. 6, pp. 24–31, June 2014.

[34] M. Backes, S. Bugiel, S. Gerling, and P. von Styp-Rekowsky, "Android security framework: Extensible multi-layered access control on android," in Proceedings of the 30th Annual Computer Security Applications Conference, ser. ACSAC '14. New York, NY, USA: ACM, 2014, pp. 46–55.[Online].Available:http://doi.acm.org/10.1145/2664243.2664265

[35] A. Reina, A. Fattori, and L. Cavallaro, "A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors," in Proceedings of the 6th European Workshop on System Security (EUROSEC), Prague, Czech Republic, April 2013.

[36] Y. Zhauniarovich, G. Russello, M. Conti, B. Crispo, and E. Fernandes, "Moses: Supporting and enforcing security profiles on smartphones," Dependable and Secure Computing, IEEE Transactions on, vol. 11, no. 3, pp. 211–223, May 2014.

[37] H. Gascon, F. Yamaguchi, D. Arp, and K. Rieck, "Structural detection of android malware using embedded call graphs," in Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, ser. AISec '13. New York, NY, USA: ACM, 2013, pp. 45–54. [Online]. Available: http://doi.acm.org/10.1145/2517312. 2517315

[38] Y. Aafer, W. Du, and H. Yin, "Droidapiminer: Mining apilevel features for robust malware detection in android," in Security and Privacy in Communication Networks, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, T. Zia, A. Zomaya, V. Varadharajan, and M. Mao, Eds. Springer International Publishing, 2013, vol. 127, pp. 86–103. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-04283-1 6

[39] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, ¨ and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket," in Proc. of NDSS, 2014.