Defending against Malwares: “Sandbox Detection and Prevention of Malwares in Android Devices”

Babysyla.L
Assistant Professor
Department of Computer Applications
College of Engineering, Trivandrum, India

Abstract— Now a days, Android malwares are growing rapidly due to its wide popularity and openness. Malware authors inject malicious code into app and upload it into Play store or third party markets. Once it is installed in the Android device, it may cause severe threats such as financial loss, privacy leakage to users etc. Therefore, this project aims at implementing an Android application sandbox system with the intent to provide an initial understanding of the behavior of unknown packages through analysis during runtime. This project is carried out as part of M. Tech Thesis work in Cyber Forensics, CDAC, Trivandrum. The proposed system executes the app under test in a sandbox environment and gives a detailed report about its behavior during runtime. In addition, a system is developed which can prevent applications from leaking privacy-sensitive data by restricting the categories of data an application can access.

Key words: Threat, Static Analysis, Sandboxes, Data Leakage, Monitoring API Calls, Restrict Permissions, Fake Data

I. INTRODUCTION

The numbers of Android users are increasing day by day due to third party apps provided by Android developers on the market. However with an increasing number of users, an increasing number of security threats targeting mobile devices [6] have also emerged. The users are more attracted towards Android platform because of the fact that these apps are freely downloadable, having most of the useful feature that the user needs in day to day life, such as free messaging, social networking, entertainment, etc. This has made Android a real target for many attackers and has resulted in rise of malicious app. Although android provides a permission mechanism [18] to restrict an access to system information or user’s information and through this feature it tries to minimize the damages that could be caused by the malicious apps. The malware writers takes an advantage of the fact that the user doesn’t understand the permissions requested by application, will grant all permission at installation time in order to use that application and tries to misuse the system by requesting dangerous permissions. The malicious activity ranges from the stealing user’s private information and sending on the internet to signing/subscribing the users to send SMS to premium numbers, etc.

In order to protect the user getting affected from this malicious activities, a secure system is needed which would help the user in identifying and preventing those malicious/harmful applications ,protecting user’s personal information and thereby securing android phone. Hence the proposed work attempts to identify such harmful applications by analysing the application in an isolated sandbox environment and generates alert/report to user. This would be helpful in preventing malicious activities by restricting the privacy of Android applications. Need to create these components, incorporating the applicable criteria that follow.

II. BACKGROUND

Before discussing the details of framework, it is important to understand how Android and Android applications work. In this section, a short introduction into the Android architecture is explained. It starts with a high level overview of the Android system architecture. And briefly explains how Android malware takes advantage of the Android platform.

A. Android System Architecture

Android is a mobile operating system developed by Open Handset Alliance (OHA) [34]. Android is implemented as a software stack, [33] customized for mobile devices. Figure 1 shows some of the most important components of this stack.

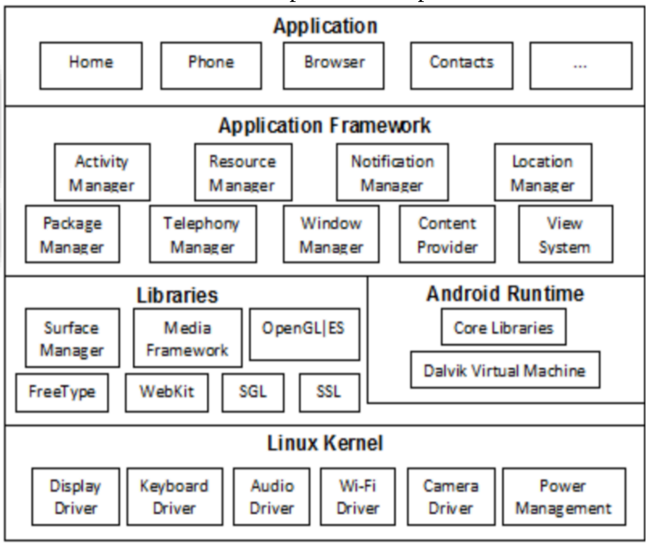

Fig. 1: Android Architecture

The core of the Android platform is a Linux kernel. The kernel’s responsibility is handling device drivers, resource access, memory-, process-, and power management and other typical OS duties. The libraries layer contains native libraries such as libc or OpenGL so as to support higher application layers. The runtime layer consists of the Dalvik Virtual Machine (DVM) with various runtime libraries. DVM is a special Java virtual machine to execute Android apps. Application framework layer contains basic services to provide activity management, SMS management, etc. Lastly, all Android apps are running on top of these layers.

B. Android SDK

Android apps are mostly written in Java using the Android SDK [32], and DVM is responsible for interpreting and executing these apps. The Android software development kit supports an emulator to run, debug and test end-user developed applications. The emulator mimics most of the features of a real device except some limitations regarding camera and video capture, headphones, battery simulation and Bluetooth. The emulator is based on QEMU which enables several operating systems to be executed on one machine and under different architectures. In this case the emulator runs an Android Linux version on an ARM simulated processor. The SDK also contains several tools to assist developers, the ones significant for this thesis are:

- Android: manages virtual devices (AVDs), projects and installed components on a SDK.
- Monkeyrunner: provides an API to programmatically control a Android device or emulator from outside of Android system.
- Android Debug Bridge (ADB): tool [33] to enable communication with an emulator instance. This can be used to install applications to the emulator and transfer files to or from the emulator or device. Another feature is the possibility to issue command-line options to the operating system through a shell interface.
- Logcat: provides a mechanism for collecting and viewing all logs issued within the emulator by the Android system and applications.

C. *Binder Mechanism*

Binder is a specialized inter-process communication (IPC) mechanism in Android. Since apps are running in their own DVM sandboxes, they need to communicate through the Binder so as to utilize others' services. Figure 2 shows the basic flow of the Binder mechanism.

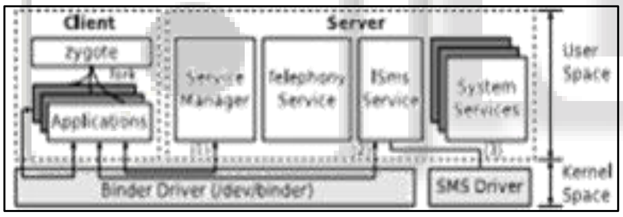

Fig. 2: Binder Mechanism

For instance, if an app wants to send an SMS message, it should first (1) contact the Service Manager which contains the information of all registered services. The Service Manager will provide a handler to communicate with the ISms Service which is responsible for sending SMS messages. Once the app has the handler, it can (2) ask the ISms Service to send SMS messages, then (3) the ISms Service will process the request and send the message through the SMS Driver. Note that all communications have to go through the Binder by sending transactions with the required information (i.e., the parcel). Transaction is a communication procedure between two processes. In Android, the Binder transaction is used to send service request (which is represented by a transaction code) to the corresponding processes. There are two stages to complete a transaction. First, the Binder will deliver a data parcel to the destination process containing the receiver information (i.e., transaction descriptor). Secondly, after completing the request, the received process will save the result in a reply parcel. In the above example of sending an SMS message, there are two transactions and they are completed in three steps. The first transaction requests for the ISms Service handler. The second transaction requests for sending an SMS message. In the second transaction, the data parcel contains the ISms Service descriptor (com.android.internal. telephony.ISms) and information (e.g., destination address and text content) for sending an SMS message. In the third step, the ISms Service sends the request and saves the result in a reply parcel.

III. METHODOLOGY

A. *Sandboxing*

The process of the analyzing app uses sandboxing. Sandboxing is based on both static and dynamic analysis components complementing each other. At first, samples go through a static pre-check. And then we extracts the basic package behavior before executing the application such as permissions, services, broadcast receivers, activities, package name, and SDK version from the manifest etc. We use the gathered information in dynamic analysis phase later. It assists in automating the dynamic analysis as well as identifying permissions, which are dangerous or commonly used by malware. Furthermore, this gives us an idea on how many permissions are requested by the app in the first place, compared to which permissions are actually used to implement the app's functionality.

The main component of the Sandbox is an emulator, that is running on the host OS containing functionality to detect data leaks and monitoring API calls, which executes application in an isolated environment .Android Emulator provided by Google is used for this purpose. It is A QEMU based emulator and mimics all the hardware and software functionality .It supports full system emulation for the ARM architecture. During execution the emulator, in this case the guest OS, broadcasts logs that are both system wide and logs that are triggered at various events during execution. Logs that are broadcasted from the emulator are intercepted using the SDK tool logcat. Since the emulator has limited logging capability hence it is patched with additional logging components for detecting malicious activities which is as follows.

1) SMS & Call monitoring: log SMS messages sent and phone calling to detect financial charge.
2) Data leakage: can be detected using dynamic taint tracking method.
3) API Monitor: Most of the malwares use specific API's for doing malicious activities. For e.g.: Malware may read, open, extract sensitive information and sent through the network.

B. *Methodology for preventing malwares*

In order to install an Android application, users are commonly required to grant these application both the permission to access information on the device, some of which users may consider private, as well as access the network, which could be used to leak this information. So two privacy controls can be used to empower users to protect their data from exfiltration by permission-hungry applications:

- Substituting shadow data in place of data that the user wants to keep private, and
- Blocking network transmissions that contain data the user made available to the application for on-device use onlyMaintaining the Integrity of the Specifications

IV. Proposed System for Malware Detection

The figure 3 depicts the proposed system and steps for detecting malwares using a sandbox:

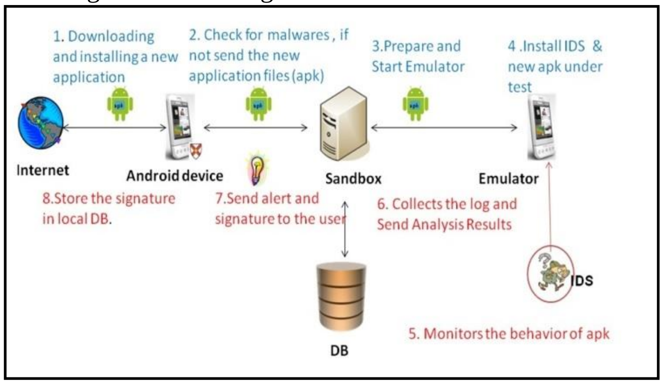
1. Downloading and installing a new application
2. Check for malwares, if not send the new application files (apk)
3.Prepare and Start Emulator
4 .Install IDS & new apk under test
Internet
Android device
Sandbox
Emulator
8.Store the signature in local DB.
7.Send alert and signature to the user
6. Collects the log and Send Analysis Results
IDS
DB
5. Monitors the behavior of apk

Fig. 3: Proposed System for Android Malware detection

The users may download and install applications from Google play or from third party market stores. At first, the newly downloaded application can be verified using signature detection module for any known malwares. If the signature is found, users can uninstall the application from mobile device. If the user wants to know the behaviour of app, it can be sent to sandbox for detailed analysis, where the app is executed in an isolated virtual environment. In the proposed system, the sandbox acts as a guest manager for controlling the Android Virtual Device and managing the execution of test sample. The sandbox pushes detection framework and necessary modules into the AVD for monitoring and detecting malicious activities. The logcat output from the AVD are collected by the sandbox and generates a report accordingly. In addition it sends an alert to user about malicious activities. Once the malware is detected, the finger print of test sample is stored in the Android Phone so that the same app need not be send for further analysis.

A. *Design of Sandbox System*

This section describes the core modules of the malware detection system and is shown in Figure 4.

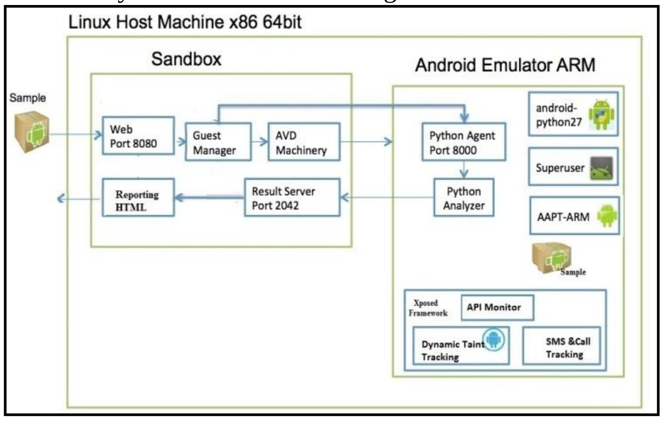

Fig. 4: System design

As defined by Wikipedia, "in computer security, a sandbox is a security mechanism for separating running programs. It is often used to execute untested code, or un trusted programs from unverified third-parties, suppliers, un trusted users and un trusted websites." This concept applies to malware analysis' sandboxing too: our goal is to run an unknown and untrusted application or file inside an isolated environment and get information on what it does. Malware sandboxing is a practical application of the dynamical analysis approach: instead of statically analysing the binary file, it gets executed and monitored in real-time. This approach obviously has pros and cons, but it's a valuable technique to obtain additional details on the malware, such as its network behavior. Therefore it's a good practice to perform both static and dynamic analysis while inspecting a malware, in order to gain a deeper understanding of it.

B. *Proposed system for prevention of malwares*

The malware detection system can be extended to prevent applications from leaking privacy-sensitive data by restricting the categories of data an application can access. It feeds applications fake data or no data at all. It can restrict several data categories, such as *contacts* or *location*.For example, if you restrict an application's access to contacts, that application will receive an empty contacts list. Similarly, restricting an application's access to your location will send a fake location to that application.

The proposed privacy manager app doesn't revoke or block permissions from an application, so most applications will continue to work as before and won't force close (crash), There are two exceptions: access to the internet and to external storage (typically an SD card)are restricted by denying access(revoking permissions).There is no other way to restrict such access because Android delegates handling these permissions to the underlying Linux network/file system. The following figure 5 depicts the malware prevention architecture.

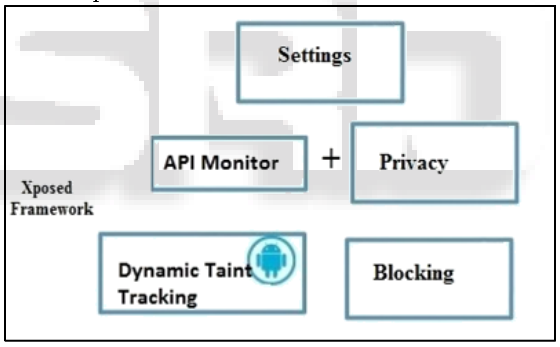

Fig. 5: Privacy manager app for restricting permissions of app

1) *Additional Components:*

a) Privacy Module

When a particular API call is encountered, it checks whether the access is restricted to application. If so, gives the shadow data to application.

b) Blocking Module

It gets the alert from Taint Tracking module and blocks network access of application.

c) Settings Module

Using this module users can construct fake data which will be sent to the application when an application has been restricted the particular data category.

C. *Design of Privacy manager system*

Since restricting a category of data for an application causes functional limitations, the new system can once again allow access to the data category to solve the issue. There is a convenient on/off switch for all restrictions for each application. By default, all newly installed applications cannot access any data category, which prevents a new application from leaking sensitive data right after installing it. Shortly after installing a new application, the proposed system will ask which data categories you want the new application to have access to. Users are allowed to edit all of an application's data categories. To help you identify potential data leaks, the new system monitors all applications' attempts to access sensitive data. It displays an orange warning triangle icon when an application has attempted to access data. If an application has requested Android permissions to access data, it displays a green key icon. It also displays an internet icon if an application has internet access, which clarifies that the application poses a risk of sharing data with an external server.

Three different options are provided for applying restrictions to privacy of app.(a)To block or control a single permission across all apps.(b) To block all permissions for a specific app (c) The app will be prompted each time an app wants to use a permission.

V. Experimental Setup and Results

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper. Sandbox is a central management software for sample execution and analysis. Each analysis is launched in a new and isolated virtual machine. The infrastructure is composed of a host machine (the management software) and a number of guest machines (AVDs that perform the analysis).The host runs the core component of the sandbox that manages the entire analysis process, while the guests are the isolated environments where the malware samples are executed safely and then analyzed.

Each guest comprised of linux machine that run android emulator, which is controlled by the machinery module.

Additional components installed inside the linux machine to support the analysis process are

- Agent.py - Python script for communicating with the machine.
- Linux analyzer component that is sent to the guest machine at the beginning of the analysis and controls the emulator.
- AAPT Arm - Android asset packaging tool compiled to arm for extracting the main activity and package name from the APK.
- ADB - android debug bridge binary for communicating with the emulator.

All modules are developed in Xposed framework which is a framework for modules that can change the behavior of the system and apps without affecting any APKs. Additional components installed inside the android emulator to support the analysis process are:

- API monitor- Dalvik API call monitoring module.
- Dynamic Taint tracking module – which uses Taintdroid interface library for adding sources and sinks.
- SMS and Call Tracking Module – which tracks the phonecalls and SMS sent by the application.

A. *Testing Environment*

Android SDK is a strict requirement for the proposed system. We can download the latest SDK from the official website.

1) *Steps for testing sandbox*

a) After downloading the sdk, go to the folder containing the .tgz fileand do the following:

```
$ tar -xvf android-sdk_r24.0.2-linux.tgz
$ cd android-sdk
$ tools/android
```

b) In the Android SDK Manager, install the following components:

- Android SDK Tools
- Android Platform-tools Tools
- newest Android SDK Tools
- Android 4.1.2 (API 16)
- SDK Platform
- ARM EABI v7a System Image

c) Add the android SDK tool to $PATH variable:
```
$exportPATH=$PATH:sdk_path/tool:sdk_path/buildtools/x.x.x.x/:sdk_path/platform-tools
```

d) Create Android Virtual Device

- Start the Android Virtual Device Manager:
- $ android avd
- Press Create..and add the following configurations:
- AVD Name - aosx
- Device - Nexus One
- Target - android 4.1.2
- Cpu/Abi - arm
- Ram - 512mb
- Vm Heap - 32
- Internal Storage - 512mb
- Sdcard size - 512 mib
- Emulation options - use host GPU
- and click OK.

e) Start analysis of app in the emulator using the script./startemu.sh avd_name

f) Run the analyser component analyser.py

g) Collect the logs and generate report using report.py script

h) Send the alert to user and store signature in the android device.

2) *Steps for testing the privacy manager app*

a) Root the Android phone/emulator.

b) Install Xposed framework and enable the modules in it.

c) Lauch the application and it will list all the installed applications in the phone.

d) There are three main approaches to controlling permissions in proposed app.

- Block a specific permission: First is to block or control a single permission across all apps. Tap the filter permissions drop down and pick a permission you want to control. For example, Contacts You'll now see all the apps installed that have permission to access your Contacts. If you want to prevent an app from having this permission tap the box to the right of it. Next time the app wants to access your address book it will be sent a empty list instead of your actual addresses.
- Block all permissions for an app: The next method for controlling permissions is to block them all for a

specific app. Select All from the permissions drop down again to view all your apps. Alongside each app are two boxes. Tapping the left-most one will selectively block all permissions for that app. It does it selectively as some permissions are not safe to block, but the important ones will be controlled

– Prompt for each permission request: By ticking the right-most box next to an app you will be prompted each time an app wants to use permission. This sounds like the best option, but in fact for a while you will be overloaded by prompt requests each time you launch an app. By default your choice will be remembered so you'll only be prompted once for each app.

VI. Conclusion

Automated malware analysis systems (or sandboxes) are one of the latest weapons in the arsenal of security vendors. Such systems execute an unknown malware program in an instrumented environment and monitor their execution. While such systems have been used as part of the manual analysis process for a while, they are increasingly used as the core of automated detection processes. The advantage of the approach is clear: It is possible to identify previously unseen (zero day) malware, as the observed activity in the sandbox is used as the basis for detection. The challenge was not to build a sandbox, but rather to build a good one. Most sandboxes leverage visualization and rely on system calls for their detection. This is not enough, since these tools fundamentally miss a significant amount of potentially relevant behaviors. Instead, I believe that a sandbox must be an analysis platform that sees all instructions that a malware program executes, thus being able to see and react to attempts by malware authors to fingerprint and detect the runtime environment.

Android app permission requests are getting out of control. More and more apps want access to your data, your location and other identifiable and highly valuable information. Fortunately, if you have rooted your phone or tablet, you can bring permissions under control. Privacy manager app offers two different approaches for protecting sensitive data from today's Android applications: shadowing sensitive data and blocking sensitive data from being ex filtrated off the device.

Acknowledgment

I express my profound gratitude and indebtedness to my project gui Ms. B. Shameedha Begum, Assistant Professor, Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, who directed me from time to time with valuable guidance, constant encouragement and suggestions throughout the project.

I express my deep sense of gratitude to my external project guid Mr. Satheesh Kumar. S, Senior Engineer, CDAC, Thiruvananthapuram who has been source of constant inspiration and for his timely guidance in the conduct of my project work

Last but not the least, I am thankful to the faculty and non-faculty members of Department of Computer Science and Engineering, National Institute of Technology, my friends, and my family for their support and comments that helped me in many aspects of my project

References

[1] MoutazAlazab, VeelashaMoonsamy Lynn Batten and RonghuaTian "Analysis of Malicious and Benign Android Applications" in 2012 32nd International Conference on Distributed Computing Systems Workshops

[2] T. Bl¨asing, L. Batyuk, A.-D. Schmidt, S. A. Camtepe, and S. Albayrak, "An Android Application Sandbox System for Suspicious Software Detection" in Proceedings of the 5th International Conference on Malicious and Unwanted Software (MALWARE), 2010

[3] Burguera, U. Zurutuza, and S. Nadjm-Tehrani. "Crowdroid: behavior-based malware detection system for android." In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, SPSM '11, pages 15–26, New York, NY, USA, 2011. ACM

[4] Gianluca Dini1, Fabio Martinelli2, Andrea Saracino1,2, and Daniele Sgandurra2 "MADAM: a Multi-Level Anomaly Detector for Android Malware".

[5] Takamasa Isohara, Keisuke Takemori and Ayumu Kubota "Kernel-based Behavior Analysis for Android Malware Detection" seventh International Conference on Computational Intelligence and Security

[6] Abdullahi Arabo and Bernardi Pranggono" Mobile Malware and Smart Device Security:Trends, Challenges and Solutions" 2013 19th International Conference on Control Systems and Computer Science

[7] Abela, Kevin Joshua L, Angeles, Don Kristopher E, Delas Alas, Jan Raynier P"An Automated Malware Detection System for Android using Behavior-based Analysis" International Journal of Cyber-Security and Digital Forensics (IJCSDF) 2(2): 1-11 The Society of Digital Information and Wireless Communications, 2013 (ISSN: 2305-0012)

[8] Abdelfattah Amamra, Chamseddine Talhi, and Jean-Marc Robert "Smartphone Malware Detection: From a Survey Towards Taxonomy" 978-1-4673-4879-9/12/$31.00_c 2012 IEEE

[9] Yajin Zhou Zhi Wang Wu Zhou Xuxian Jiang "Hey, You, Get Off of My Market:Detecting Malicious Apps in Official and Alternative Android Markets" International Conference on Computational Intelligence and Security

[10] Maslennikov, D. "Mobile Malware Evolution," Part 5. 2012 [cited 2012/ 26/ March], Available from: www.securelist.com/en/analysis/204792222/Mobile Malware Evolution_Part_5.

[11] Peyman Kabiri, A.A.G.," Research on Intrusion Detection and Response: A Survey " International Journal of Network Security, 2005.1(2): p. 84-102 Takamasa Isohara, Keisuke Takemori and Ayumu Kubota "Kernel-based Behavior Analysis for Android Malware Detection seventh International Conference on Computational Intelligence and Security

[12] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, Y. Weiss:" Andromaly: a behavioral malware detection framework for android devices." Journal of Intelligent Information Systems 38(1) (January 2011)

[13] Xie, L., Zhang, X., Seifert, J.P., Zhu, S.: "pBMDS: a behavior-based malware detection system for cellphone devices." In: Proceedings of the Third ACM

Conference on Wireless Network Security, WISEC 2010, Hoboken, New Jersey, USA, March 22-24, 2010, ACM (2010)
[14] D. Damopoulos, S.A. Menesidou, G. Kambourakis, M. Papadaki, N. Clarke, S. Gritzalis: "Evaluation of Anomaly-Based IDS for Mobile Devices Using Machine Learning Classifiers." Security and Communications Networks 5(00) (2011)
[15] Guangdong Bai, Liang Gu, Tao Feng, Yao Guo, and Xiangqun Chen. "Context-aware usage control for android. In Security and Privacy in Communication Networks," volume 50 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 326-343. Springer Berlin Heidelberg, 2010.
[16] Abhijit Bose, Xin Hu, Kang G. Shin, and Taejoon Park." Behavioral detection of malware on mobile Handsets". In Proceeding of the 6th international conference on Mobile systems, applications, and services, MobiSys '08, pages 225{238, New York, NY, USA, 2008. ACM.
[17] Francesco Di Cerbo, Andrea Girardello, Florian Michahelles, and Svetlana Voronkova." Detection of malicious applications on android OS" In Proceedings of the 4th international conference on Computational forensics, IWCF'10, pages 138{149, Berlin, Heidelberg, 2011. Springer-Verlag.
[18] William Enck, Machigar Ongtang, and Patrick McDaniel. "Understanding android security." IEEE Security and Privacy, 7:50{57, January 2009.
[19] Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji." Intrusion detection using sequences of system calls." J. Comput. Secur., 6:151{180, August 1998.
[20] Aubrey-Derrick Schmidt, Frank Peters, Florian Lamour, Christian Scheel, Seyit Ahmet Camtepe, and Sahin Albayrak. "Monitoring smartphones for anomaly detection. "Mobile Networks and Applications (MONET) {SPECIAL ISSUE on Mobility of Systems, Users, Data and Computing, November 2008.
[21] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, Shlomi Dolev, and Chanan Glezer."Google android: A comprehensive security assessment". IEEE Security and Privacy, 8:35{44, 2010.
[22] Desnos and P. Lantz, "Droidbox: An android application sandbox for dynamicanalysis." [Online]. Available: http: //project.honeynet.org/gsoc2011/slot5
[23] Lockheimer,"AndroidandSecurity.[Online].Available:http://googlemobile.blogspot.com/2012/02/android-and-security.html
[24] P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified,"Electrical Engineering and Computer Sciences University of California at Berkeley, Technical Report EECS-2011-48, 2011.
[25] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. "Taintdroid: an information tracking system for realtime privacy monitoring on smartphones". In Proceedings of the 9th USENIX conference on Operating systems design and implementation, OSDI'10, pages 1{6, Berkeley, CA, USA, 2010. USENIX Association.
[26] Google Inc. Android market. https://market.android.com/.
[27] C. Bonnington, "Google's 10 billion android app downloads: By the numbers." [Online]. Available: http://www.wired.com/gadgetlab/2011/12/10-billion-apps-detailed
[28] Chris Fleizach, Michael Liljienstam, Per Johansson, Geofrey M. Voelker, and Andras Mehes. "Can you infect me now? malware propagation in mobile phone networks." In Proceedings of the 5th ACM Workshop on Recurring Malcode (WORM 2007), Alexandira, Virginia, USA, nov 2007. ACM.
[29] Aubrey-Derrick Schmidt. "Anomaly detection on smartphones." Proceedings of the Third GI Graduate Workshop on Reactive Security (SPRING), number SR-2008-01 in GI SIG SIDAR Technical Reports, Stuttgart, Germany, August 2008."
[30] Zhou, Y. Zhou, X. Jiang, and P. Ning. "DroidMOSS: Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces. "In Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy, CODASPY'12, 2012.
[31] Vaibhav Rastogi, Yan Chen, and William Enck, "AppsPlayground: Automatic Security Analysis of Smartphone Applications" CODASPY'13, February 18–20, 2013, San Antonio, Texas, USA.
[32] Using the android emulator (2013). URL http://developer.android.com/tools/devices/emulator.html
[33] Android debug bridge - android developer documentation (2012). URL http://developer.android.com/tools/help/adb.html
[34] Open Handset Alliance. Android - an open handset alliance project. http://developer.android.com/, 2008.
[35] Michael Spreitzenbarth, Felix Freiling, Florian Echtler, Thomas Schreck, and Johannes Hoffmann. 2013. Mobile-sandbox: having a deeper look into android applications. In Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC '13).
[36] YuanZhang Min Yang, Bingquan, Xu,Zhemin, Yang,Guofei Gu,Peng Ning,X.Sean Wang,Binyu Zang "Vetting Undesirable Behaviors in Android Apps with Permission Use Analysis "In Proc.of WiSec'12, 2012.
[37] Peter Hornyack, Seungyeop Han, Jaeyeon Jung "These Aren't the Droids You're Looking For" Retrofitting Android to Protect Data from Imperious Applications in Proceedings of University of Washington and Microsoft Research 2014.
[38] Mingshen Sun, Min Zheng, John C. "Design and Implementation of an Android Host-based Intrusion Prevention System" in ACSAC '14, December 08–12 2014, New Orleans, LA, USA.