International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-2, Issue-5, September-2014

System Level Security Solution for Android

Aparna Bhonde, Madhumita Chatterjee

Abstract— **Android Operating System, by Open Handset Alliance, prominently led by Google is dominating the share of smart phones. Mobile applications like banking, e-shopping, business apps used on these devices have become foundational tool for today's workforce.**

However the Smartphone users are under continuous threat of exposure and misuse of their personal information due to rapid growth of malware for android which significantly exceeds that of other platforms. Android being open platform supports the development of applications. Now a day's one can publish an app after registration as a developer for USD25. Due to its availability to all android users, the android market is the main channel of malware distribution. Along with its growth, the importance of security has also risen. A proportional increase in the number of vulnerabilities is also happening to the extent that there are limited numbers of security applications available to protect these devices. Among the security apps many antivirus which work on the application layer are present in the market which claims the security.

However, the efficacies of these applications have not been empirically established. After studying the shortcomings and demerits of the available solutions, an enhanced security solution for android application assessment at the operating system level is suggested. The solution customizes the android operating system mainly the package manager which holds the notification of the activities which takes place on the device. The package manager is updated to receive the intent passed by the verification agent activity. An application is built which is hooked to the package manager which checks across the database signatures for the malwares and blocks the installation process of the android application on the system. Altogether a new android system image is compiled and tested across the known set of malwares. Unlike antivirus, this check takes place before the process of installation due to which, we are able to mitigate attacks caused by malwares on android smart phones by variety of applications.

Index Terms—**Android, smartphones, application security, malware detection.**

MOBILE computing is a fact of life in the modern enterprise. With the rapid and everyday adoption of mobile devices, enterprise applications have been extended beyond the confines of the corporate network. The large attack surface and the proliferation of mobile devices have created a significant security challenge for companies and the IT professionals. The mobile security stack consists of the Infrastructure layer, hardware layer, Operating system layer and the Application layer. Most of the attacks that are registered are device based attacks, network based attacks and the server based attacks. Out of these the most prominently occurred attacks are the device based attacks [4]. Attacks against the device are most tangible, impactful and obvious to the average person. However, a more dangerous scenario occurs when users download unknown applications or from the Android App Store. This could lead to information leakage or complete compromise of the device, allowing attackers to install malicious certificates, reconfigure proxy settings or allow man-in-middle (MiTM) visibility into every user transaction. Hence according to [5] the application layer has the largest attack surface where maximum damage to security occurs.

Gartner analysis [15] says that Android is an open source operating system, prominently led by Google, is having the maximum market share, where developers can develop their applications and make it available in the market to the users. There is a great difficulty to find out the authenticity of the applications which are downloaded by millions of people every day on their smart phones. Hence to keep a check on the malwares and the authenticity of the application we need to have such a solution which is not dependent on the third party.

Third party applications which are developed at the application layer for the assessment of any android app across malware require the system permissions from the package manager. But the fact is, package manager does not grant system permissions to any third party application, until and unless the android system is root. According to [2] [3], rooting is a process that allows attaining root access to the Android Operating system code. It gives the privileges to modify the software code on the device or install other software that the manufacturer wouldn't normally allow to do. The process of rooting makes the system vulnerable for attacks as it does not have to do anything to get to the super user.

Customizing the android operating system is different than the rooting process. Rooting is a cosmetic procedure and does not make any changes to the operating system. It only gives elevated privileges to the user-root access.

The antivirus which claims the security by checking the malwares in the installed applications on the system also fails when the malicious app spreads itself blocking the antivirus. Above all we need such a solution which checks for the malware in the android app before it gets installed on the system, Hence we suggest a solution to customize the android operating system which will scan for the malwares against the signature database. This process of scanning the app takes place before the installation of the application on the smart phone. Hence up to certain extend it mitigates the risk of the smart phone getting compromised due to malicious android apps.

Manuscript received September 23, 2014

Aparna Bhonde, Department of IT, PIIT Panvel, Mumbai University, India, 9819829042

Madhumita Chatterjee , Department of IT, PIIT Panvel, Mumbai University, India,

Our Work

The solution customizes the latest version of android operating system, KitKat. Mainly the Package manager service, present in the android operating system is customized which holds the notification of the activities. Every time as soon as a new app tries to get installed on the device, Package Manager will trigger the MalwareTest App internally and check the application across the database signatures for malwares. It gives the notification to the user about the malware and then user can block the installation process. Altogether a new android system image is complied. This facility is not present in any of the android versions till date. The solution overcomes the difficulty of the sandbox based file system as well as the android permission model. Hence, we have tried to apply a security solution on the top of android Operating system.

The main advantage of using this solution is that the apk is blocked before installation if it contains malware. This process does not toil in the background as it triggers only on arrival of apk hence less amount of power is consumed which is very important constraint to increase the efficiency of a mobile device.

The remainder of this paper is structured as follows. In Section 2 the background theory is introduced which includes Android system basics and the discussion of the security system provided by android operating system. The shortcomings of antivirus software on the Android Platform are explained in Section 3. In Section 4 we introduce our concept for an enhancement in the security of android operating system. We will discuss the implementation of the enhanced security solution for the android platform and Section 5 will be the conclusion

I. BACKGROUND THEORY

A. *Android*

Android is an operating system designed for smart phones which provide a sandboxed application execution environment. A customized embedded Linux system interacts with the phone hardware and an off processor radio. The Binder middleware and the application API runs on the top of Linux. Hence to simplify, an applications only interface to the phone is through these API's. Each application is executed within a Dalvik Virtual Machine (DVM) running under UNIX uid[10]. The phone comes pre-installed with a selection of system applications like phone dialer, address book. Applications interact with each other and the phone through different form of IPC (inter process communication).

B. *Securable IPC mechanism*

- Activity

An Activity is, generally, the code for a single, user-focused task. It usually includes displaying a UI to the user. Typically, one of the application's activities is the entry point to an application. Intents are used to specify as Activity, and this may be done ambiguously to allow the user to configure their preferred handler.

- Broadcasts

Broadcasts provide a way to send messages between applications, for example, alerting listeners to the passage of time, an incoming message, or other data. When sending a broadcast as application puts the message to be sent into intent. The application can specify which Broadcasts they care about in terms of the intents they wish to receive by specifying an Intent Filter. Broadcast is instantiated when an IPC mechanism known as an Intent is issued by the operating system or another application. An application may register a receiver for the low battery message, for example and change its behavior based on that information.

- Services

Services are background processes that toil away quietly in the background. It can run in its own process, or in the context of another application's process. Other components 'bind' to a service and invoke methods on it via a remote procedure calls. A service might play music, even when the user quits the media –selection UI, the user probably still intends for the music to keep playing and others handle incoming instant messages, file transfers or email. Services can be started using intents.

- Content Providers

Content Providers provide a way to efficiently share relational data between processes securely. They are based on SQL and should be used carefully.

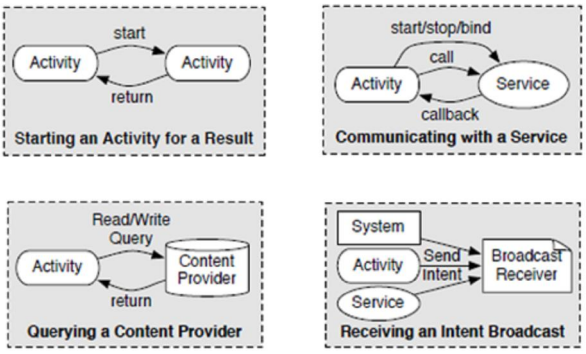

Figure 1. Android's IPC Mecahnism[17]

Content Providers can be secured with Android permissions, and used to share data between processes, like files might be on traditional UNIX like systems.

- Binder

Binder provides a highly efficient communication mechanism on Android. It is implemented in the kernel, and you can easily build RPC interfaces on top of it using the Android Interface Definition Language (AIDL). Binder is commonly used to bridge Java and native code running in separate processes.

The key security features of android to achieve the objectives like protection of user data, protection of system resources including the network and provide application isolation are as follows:

1. Robust security at the OS level through the Linux kernel.
2. Mandatory application sandbox for all applications.
3. Secure inter process communication.
4. Application signing.
5. Application defined and user granted permissions.

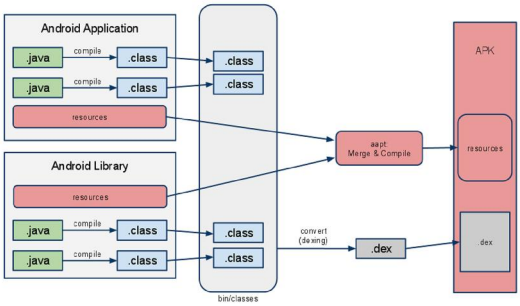

Figure 2. Android's APK File [16]

C. Installation process of Android Application

An Android Application is stored in an APK file. In order to run the app one needs to install the required APK file. An APK file consists of java class files and the libraries as shown in fig. At the time of application installation, the list of permissions [11] is asked to the user. If the user agrees to the listed permissions and clicks install then the installation takes place. There are 2 types of applications, one being from the Android Google Play store and others are third party applications. If the application to be installed is from third party then user has to enable the installation from unknown resources. The process is shown in fig

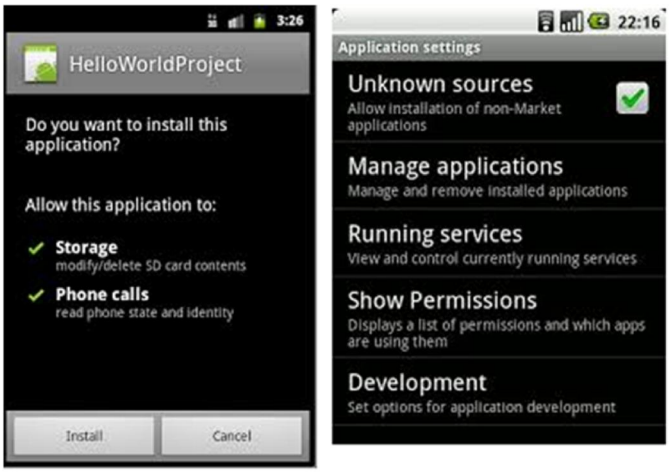

Figure 3. Android's APK installation process

System server is the core of Android system and it starts as soon as Dalvik is initialized and running. The main Android services such as the activity manager, package manager, and alarm manager are running in their separate threads but as parts of system server process. Package Manager is an API that actually manages application install, uninstall and upgrade. When an APK is installed, Package Manager Parse the package (APK) file and displays confirmation. When the user presses OK button, Package Manager calls the method named "installPackage" with these four parameters namely uri, installFlags , observer, installPackageName. Package manager starts one service named "package" which actually carries out the processing of this service. Package manager Service runs in the system service process and installs daemon (installd) runs as a native process. Both start at the same boot time.

Package installer

It is a default application for Android to interactively install a normal package. Package installer provide user interface to manage applications/ packages. Package Installer calls InstallAppProgress activity to receive instructions from the user. InstallAppProgress will ask Package Manager Service to install package via installd. Some of the main tasks of Package manager Service are add a package to the queue for the installation process, determine the appropriate location of the package installation, determine installation Install/Update new, Copy the apk to a given directory, determine the UID of the app, request the installed daemon process, create the application directory and set permissions, extraction of dex code to the cache directory.

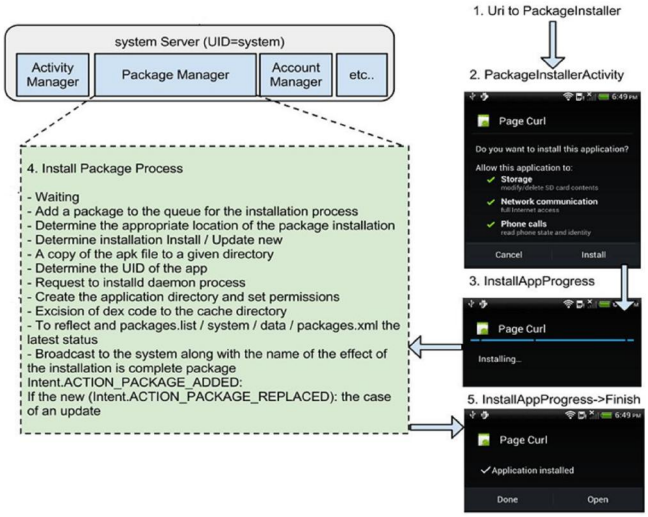

Figure 4. Working of Package Manager [3]

According to [2] the android OS only reveals only the permissions to the user. It also checks from its Google play store database whether the app is authentic or not. But the issue here is, user cannot judge just by displaying the permissions whether the app is malicious or not. Hence leads to unknowingly spreading the malware. Hence there is a need for solution at the application layer.

D. Solutions for preventing Malwares

Currently according to [6] there are many antivirus available in the market to scan the APK for malwares, but mobiles are among the resource constrained devices hence the applications need to have limited processing, low memory and operate on low power mode due to finite energy supply [12].

The major limitation in using anti-virus application is it scans the system for malwares after the installation of the Apk file. Hence it fails in case of malwares which spreads and attacks the working of anti-virus application itself.

II. LIMITATION OF ANTI-VIRUS

Mobiles are among the resource constrained devices hence the applications need to have limited processing, low memory and operate on low power mode due to finite energy supply. According to [7] the antivirus software majorly consumes the battery which reduces the performance of the

smart phones. Android Anti-Virus software is also limited drastically by file system-based sandboxing. It cannot scan the file system on demand or monitor file system changes. Most importantly, this includes the working directories of the other apps. Anti-Virus software is thus oblivious to any files other apps might download or create at runtime, including malicious code [6].Package Database in android OS keeps track of installed apps in a package database [13]. This database contains the code path where an apps package file with its byte code is stored, the apps package name, its UID and other entries. In contrast to many other android OS resources, the package database is publically readable.

Access Package files themselves are also readable by any app. This in combination with package database being readable provides access to package files. Antivirus software can acquire the path to package files from the package database and then open package files directly. This way, common antivirus detection techniques can at least be applied to the static app installation package file.

Antivirus basically works on 2 methods that are heuristic and signature based [6]. Heuristic method is to analyze the suspicious files characteristics and behavior to determine if it is indeed malware, where signature based method identify known malware saved on the database. If the virus then reappears, it can be identified as such using the signature and assigned to a specific virus. According to [6], Android antivirus cannot deploy recognition techniques based on the heuristics to arbitrary file system objects, and especially not to apps working directories contents. Thus, dynamically downloaded code will not be found. This dynamically fetched code may also be the only component which openly demonstrates malicious behavior, keeping the app which downloaded the malicious payload free of any suspicion and detection.

So Major hindrance for antivirus software is,

1. The android OS itself uses unique user IDs to create each Android process which is the concept of sandboxed applications. Hence it's unable to directly access the file system and its contents.
2. When a virus tries to modify core system files or affect other vital parts of the android device, existing antivirus software can't recognize that because it isn't able to access the root of the system.

In other words, rooting android could be the only solution to androids security problems which is not recommended due to other security issues.

III. PROPOSED MODEL

Android's source code is released by Google under the Apache license, this permissive licensing allows the software to be freely modified by users. Android device owners are not given root access to the operating system and sensitive partitions such as /system is read-only. However, root access can be obtained by exploiting security flaws in android, which is used frequently by the open-source community to enhance the capabilities of their devices.

In our proposed system we are trying to develop a third party application which requires system permission to hook the package manager. In Android OS the package manager has defined some protection levels for the permissions, which are grouped on

1. Regular, a lower-risk permission that gives requesting access to isolated application level features with minimal risk to other applications the system or the user. The system automatically grants this type of permission to a requesting application at installation, without asking for the user's explicit approval.
2. Dangerous, a higher-risk permission that would give a requesting application access to private user data or control over the device that can negatively impact the user. Because this type of permission introduces potential risk, the system may not automatically grant it to the requesting application. For example any dangerous permission requested by an application may be displayed to the user and require confirmation before proceeding or some other approach may be taken to avoid the user automatically allowing the use of such facilities.
3. Signed is a permission that the system grants only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval.
4. Signatures or System is a permission that the system grants only to applications that are in the android system image or that are signed with the same certificate as the application that declared the permission. Please avoid using this option , as the signature protection level should be sufficient for most needs and works regardless of exactly where applications are installed. This permission is used for certain special situations where multiple vendors have applications built into a system image and need to share specific features explicitly because they are being built together.

According to [8][9], Permissions in the first two groups can be granted to any application, where as the last two can be obtained only by applications which are system preinstalled in the device's firmware or which are signed with the platform key, i.e. the same key that was used to sign the firmware.

Fact is Package Manager does not grant system permissions to hook until and unless the android system is root. It gives the privileges to modify the software code on the device or install other software that the manufacturer would not normally allow to. Hence for good mobile security reasons they don't want users to make modifications to the phones that could result accident beyond repair.

Android users are restoring to them because of the powerful perks they provide, such as:

1. Full customization for just about every theme/graphic
2. Download of any app, regardless of the app store they are posted on
3. Extended battery life and added performance.
4. Updates to the latest version of Android if your device is outdated and no longer updated by the manufacturer.

To secure the Android operating system from the malware attacks, the APK should be scanned before it gets

installed on the android operating system. Hence to achieve this, we need to customize the operating system as the development needs to be in root. Hence the proposed model requires cooking the device firmware and adding a custom package verification agent into the firmware. Next we need to add an activity into that agent on package manager which will generate the checksum using SHA algorithm [14] and checks across the database for malicious signature. Finally, if the signature is found in the database then the installation process is blocked and if the application is without any malwares the installation process is carried out by giving result to the package manager. The block diagram depicts the flow of the proposed model.

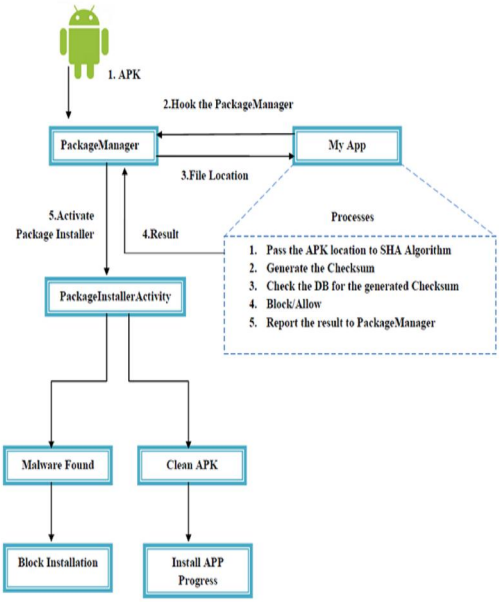

Figure 5. Block diagram of Proposed Model

About the Serverside dependencies the system works on a thin client totally and has very few serverside dependencies. At the server side the database is developed by applying SHA algorithm for the known malwars. The application needs to get updated with this database at a regular time interval. The objective of the above model is to

1. Secure the Android OS from malware before apk file installation takes place.
2. Block the installation process if the app is malicious.
3. Improve the performance of the resource containt device by triggering the application only at the time of installation of the APK file.

IV IMPLEMENTATION AND TESTING

The above model was tested for some of the latest malwares families. For this purpose the following test cases were designed.

Test case 1: Detection of unaltered malware: In this test case the malware application package files are pushed on the SDcard and checked. The above model was tested for the latest malwares families which successfully detects the known malwares.

Test case 2: Detection of altered malware: In this test case the malware application package files are decompiled and their package and class names were renamed but no code is altered.

Test case 3: Dynamic downloading: In this test case an app is directly allowed to download and checked for other dynamic infection routines.

Test case 4: Advanced and unknown threats: In this test case some of the latest malicious files were tried to test across the database signatures.

The results show that the model is significantly detecting the malwares except for some samples in Test case 2 as application package files were decompiled and their package and class names were renamed which resulted in new SHA-1 checksum, which was not present in the signature database. Regarding the Test case 4, the result was failed because latest malicious file signatures were not updated.Overall out of 26 samples tested 23 were detected for the 3 test cases and zero were detected for the test case 4.The following fig.7 shows the screen shots of the actual tests conducted and the results are shown in the table below.

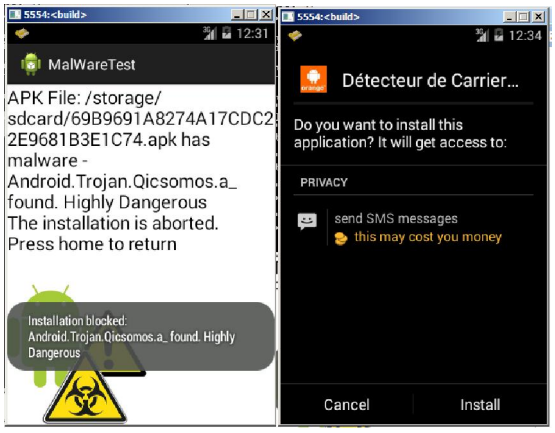

Figure 6: Screenshot of malware detected

This model hence trying to include more security at the operating system level.

Sr no.	Malware Family	Samples	Description	Impact	Test case 1	Test case 2	Test case 3
1	AndroidFakeToken	2	Steals information and sends it to certain remote servers	Low	Pass	Pass	Pass
2	DroidKungFuUpadte	4	Sends sensitive information to an attacker and includes backdoor function-ality. It also exploits vulnerabilities to gain root access	Medium	Pass	2 Pass 1Fail	Pass
3	FakeInst	2	Send SMS messages to premium-rate numbers or services.	Low	Pass	Pass	Pass
5	JiFake	2	Send SMS messages to premium-rate numbers or services.	Low	Pass	Pass	Pass
6	OpFake	4	Monitors SMS messages and is capable of deleting/moving messages based on the originating phone numbers and message content.	High	Pass	Pass	Pass
7	Plankton	6	Silently forward information about the device to a remote location as well as download additional file on the device	High	Pass	3 Pass 3Fail	Pass
8	SpitMo	4	Steals credentials mainly related to banking apps	High	Pass	Pass	Pass
9	Zitmo	3	Steals telephone number, device information and incoming SMS messages.	High	Pass	Pass	Pass

IV. CONCLUSION

Due to androids secured IPC mechanism antivirus software is not very effective on android platform. However there is significant increase in the growth of malwares, hence an effective malware detection technique above the android operating system adds one more layer of security. Our contribution to this area is detecting the malware on the android system before it gets installed, due to this detection technique the possibility of spreading the malware after installation is totally ruled out. Traditional signature based detection is impemented in this model and we look forward to some more advanced detection techniques which could work hand in hand with the current sandbox based file system limitations. This process does not toil in the background as it triggers only on arrival of apk hence less amount of power is consumed which is very important parameter to increase the efficiency of a mobile device.

In a nutshell, an attempt for mitigating the malware attacks with achieving efficiency on the widely used operating system, android is done. Results show that we were able to detect malware apks and prevent them from getting installed. Testing this model across many more malware samples will be part of our future work.

REFERENCES

[1] Fraunhofer AISEC, Android OS Security: Risks and Limitations, A Practical Evaluation Version 1.0, by: Rafael Fedler, Christian Banse, Christoph Kraub and Volker Fusenig.

[2] University of California ,Berkeley, Android Permissions Demystied, by: CCS11, October 1721, 2011, Chicago, Illionois USA Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, David Wagner apf, emc, sch, dawnsong, daw@ cs.berkeley.edu

[3] Package Manager In Depth, In Depth: Android Package Manager and Package installer, http://java.dzone.com/articles/depth-android-package-manager

[4] Mobile Security, Identifying the Mobile Security Stack, http://blog.veracode.com/2011/03/identifying-the-mobile-security-stack/

[5] Reference Architecture, Mobile Security Reference Architecture, by CIO Council and US Department of HomeLAnd Security May 2013, https://cio.gov/wpcontent/uploads/2013/05/Mobile-Security-Reference-Architecture.pdf

[6] 2013 IEEE, An AntiVirus API for Android Malware Recognition, by Rafael Fedler, Marcel Kulicke and Julian AISEC Garching near Munich, Germany rafel,kuliche.schuette@aisec.fraunhofer.de http://ieeexplore.ieee.org(PISNumber3A6703677)

[7] Review August 2013, AV-Comparatives Mobile Security, http://www.av-comparatives.org/wp-content/uploads/2013/08/avc_mob_201308_en.pdf

[8] Permission, Android Development Guide, http://developer.android.com/guide/topics/security/permissionas.html

[9] Permission Element, Android development Guide, http://developer.android.com/guide/topics/manifest/permission-element.html

[10] Systems Internet Infrastructure Security Laboratory. Department of Computer Science and Engineering. The Pennsylvania State University, A Study of Android Application Security, by William Enck, Damien Octeau, Patrick McDaniel and Swarat Chaudhari http://dl.acm.org/citation.cfm?id=2028088

[11] Symposium on Usable Privacy and Security (SOUPS) 2012, July 11-13,2012, Washington, DC, USA, Android Permissions: User Attention, Comprehension and Behavior by: Adrienne Porter Flet, Elizabeth Hay, Serge Egelman, Ariel Haneyy, Erika Chin, David Wagner Computer Science Department School of Information. University of California, Berkeley

[12] Smart Devices, Ubiquitous Computing Smart Devices, Environments and Interactions by: Stefan Poslad Queen Mary, University of London, UK

[13] J.Burns. Exploraory Android Surgery (talk slides), Black Hat Technical Security Conference USA, May 2009. http://www.blackhat.com/html/bhusa-09/bhusa-09-archives.html

[14] NIST, Descriptions of SHA-1 http://csrc.nist.gov/groups/ST/toolkit/documents/Examples/**SHA1**.**pdf**

[15] Gartner research Smartphone sales operating system http://www.gartner.com/newsroom/id/2665715

[16] Android APK Architecture, http://devmaze.wordpress.com/2011/05/22/android-application-android-libraries-and-jar-libraries/

[17] Systems and Internet Infrastructure security, Pennsylvania State University, Understanding Android's security framework by: William Enck, Patrick McDaniel