IJST EXPLORE International Journal of Science and Technology
CENTRE OF PROFESSIONAL RESEARCH PUBLICATIONS
ISSN 2049-7318

International Journal of Science and Technology Volume 2 No. 11, November, 2013

Qualitative Evaluation of Security Tools for Android

F. Tchakounté, **P. Dayang**
The University of Ngaoundéré, Faculty of Science, PO Box 454 Ngaoundéré, Cameroon

ABSTRACT

Android has become the most popular operating system for Smartphone, which renders it as an attractive target for malwares. Several techniques from literature have been developed to protect sensitive information of users from being stolen. Unfortunately, until now these techniques are not sufficient and therefore need to be improved. These techniques mainly use some tools already developed to disassemble or to study behaviour of malicious programs. Additionally, there exist integrated environments that people use to perform isolated analysis in virtual machine. We conduct in this work a qualitative study of these tools, based on the criteria such as documentation, usability, functional, portability, security, and extensibility. We found that tools rarely have these six quality characteristics and this lack gives the attacker the opportunity to take advantage of the user's information. In addition, we found that tools used for analysis can be entry doors to attacks. We identified possible vulnerabilities and propose ways to mitigate them further on.

Key words: *smartphone, security, analysis, Android, security tool*

1. INTRODUCTION

Android has become the most popular operating system for smartphones [6]. These mobile and versatile devices offer interesting functionalities to the users such as Internet, Camera, GPS tracking, and diverse other useful applications. The development of applications for Android OS grows with its popularity. We have more than 450,000 apps in the official Google Play market and more than 850,000 activations of new devices per day [31]. However, applications manipulate sensitive information targeted by attackers such as password, personal accounts, contacts, SMS, etc. While Android smartphones are convenient to users, they are attractive to malicious developers as they contain data extremely interesting to hackers. For instance, over 250.000 Android users were compromised when they downloaded malicious software disguised as legitimate applications from the Android Market [7]. In the meantime, techniques used by malware developers evolved and become more sophisticated. Zhou and Jiang [3] classify malware installation into three social engineering-based techniques consisting to mislead users into downloading malicious apps unknowingly: repackaging, update attacks and drive-by download [7]. Once installed, malwares rely on the built-in support of automated event notification (BOOT_COMPLETED, PHONE_STATE) to trigger or launch its malicious payloads. Android platform vulnerabilities can be exploited by malwares to facilitate the execution of their payload. Malwares can use HTTP-based web traffic to receive *bot* commands from remote servers, or they can collect user's information to upload to a remote server.

Moreover, Android security is based on isolation access control. The access to personal information has to be explicitly granted at installation time: during the installation of an app the user is provided with a list of permissions the app requests. The user can either accept all of these permissions, or refuse them. In the first case the app will be installed and in the second, the app will not be installed. Users can neither dynamically grant and revoke permissions at runtime, nor add restrictions according to their personal needs. Additionally, the user often is not aware of permission requirements necessitated by the app. This is a real limitation of Android permission system [32].

In research, malware analysis techniques help to understand unpredictable behaviour of malwares and permissions over used as payload for malwares. Malware analysis includes two complementary mechanisms: dynamic analysis and static analysis.

The first one consists of executing a given sample in an isolated environment to monitor its behaviour and determine whether it is malicious and what changes are incurred in the system. The second one decompiles the .apk file and transforms it into the .java source code [4][5].

The goals and contributions of this paper are three-fold.
First, we fulfil the need to present the entire malware analysis ecosystem. We propose and justify some security criteria important to evaluate a good tool for mobile analysis.
Second, we characterise dynamic and static analysis tools provided by SecMobi Wiki [8] using own defined criteria. Based on our criteria, we highlight some advantages and disadvantages of each type of analysis. Additionally, we characterise environments for isolated analysis and revealing positive and negative sides likewise. These elements can guide one to choose a tool.

Third, we perform a qualitative security evaluation using the selected tools and propose recommendations for security challenges.

2. MALWARE ANALYSIS ECOSYSTEM AND METHODS

Today, malwares do not target Desktop systems only, but mainly mobile systems. They play a part in most security

incidents. *Malware analysis* is the art of dissecting malware to understand how it works, how to identify it, and how to defeat or eliminate it [33].
The objective of *malware analysis* is to provide the information needed to respond to an intrusion. Some objects are taken in account during the malware analysis process (Figure 1):

The software: It is submitted to analyse in a *host* to see whether it does something that causes harm to a user, computer, or network. In the positive case according to *results* and *decisions*, it can be considered as *malware* including viruses, trojan horses, worms, rootkits, scareware, and spyware.

Tool for analysis: Generally, it is the software built to examine un-trusted applications.
Result: The result is the output of the tool analysis. Some results can be the changed state of the file system, deleted sensitive files. Based on these results, malwares can be classified.

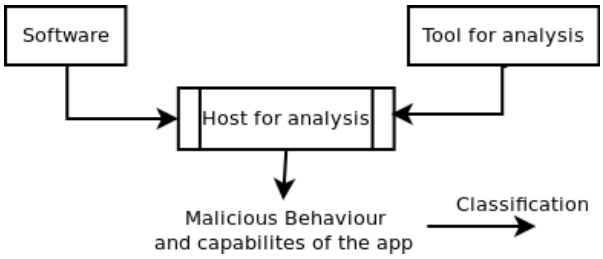

Figure1: malware analysis ecosystem

2.1. Samples

Table 1 lists tools [8] that we use in this paper for evaluating. There are classified into three categories: Integrated Analysis Environments, tools used for static and dynamic analysis. Our study was guided entirely with the SecMobi Wiki findings [8]. The choice of these tools is based on (a) the fact that they are mainly used in literature; (b) they are open source, and (3) our experience of using them.

2.2 Evaluation criteria

Here, we outline criteria we use to evaluate tool samples. These characteristics have positive incidence of the success of the analysis of malwares. The evaluation criteria for each sample are collected from the respective official Websites.

Usability (U): The tool for analysis should provide a user interface (UI) that is easy to use for the one who wants to perform an analysis. The question to answer is "Is the malware analysis tool easy to use?"

Table1. Samples for evaluation

Sample category	Names
static analysis [34]	smali/baksmali, Dedexer, radare, dex2jar, ded, jd-gui/jad, apktool, AXMLPrinter
dynamic analysis [35]	Taintdroid, DroidBox, LogCat, Strace, Mobile Sandbox, Anubis
Integrated environments [36]	Santoku, ARE, Mobisec, Tamer, Androguard, Apkinspector

Documented (D): The tool for analysis is a software. Therefore, it should be well documented for the user to understand the objectives, and features. The question to answer is "Does the malware analysis tool provide sufficient and understandable documentation to learn how to use it?"

Functional (F): When one wants to analyse, it is important to know if the chosen tool has required features. This element depends on the previous criteria. The question here is "Does the malware analysis tool provide enough functionalities to learn the behaviour of malware explicitly?"

Portability (P): The tool should work on several platforms. The question to answer is "Does the analysis tool work on different platforms?"

Extensibility (E): The malware analyst should be able to adapt the tool to study the malware behaviour accordingly. The question to answer is "Can the analysis tool be modified to integrate other functionalities?"

Security (S): The tool is an application aiming to track malwares. This is a sufficient reason to be targeted by malware attackers. The question here is "Does the analysis tool enough secured to prevent being compromised". We will insist on this criterion in Section 3.4.

3. RESULTS AND DISCUSSIONS

3.1 Evaluation of Reverse Engineering Tools

This technique is a fast and inexpensive approach in terms of resource constraints to find malicious characteristics or bad code segments in an application from the binary without installing. Table 2 shows that 88,88% of tools from the first category are based on command line. Because of the memorization and familiarity needed to operate a command line interface, inexperienced users have difficulties to navigate and operate command line interface. The user should know very well the command meanings to proceed, in order to expect some results. Considering the security benefits, command line has simple design and fewer targets of bugs and vulnerabilities than graphical user interface. Even, documentation that accompany these software is not easy understandable for a beginner user. User must be closed to Scripting languages, Java language, Android platform, shell commands, and system operations such as building from source code, installing, and running. This takes a lot of time and effort to practice then to transfer to real smartphone device. Whereas some tools are provided with good documentation, some are not (*AXMLPrinter*), [18]. The following table classifies tools according to the aforementioned evaluation criteria. This has an advantage in the sort that the user choice is well oriented. User skills play an important for the choice of the appropriate tools. Unfortunately, our static security analysis samples do not fit

all with the aspect of platform dependency, which limits the choice scope. In fact, the majority of tools are turned to Linux kernel and require the presence of JDK (Java Development Kit). In some cases, the testing is not yet finished, for instance, the *Dare* tool [13]. This concurs to augment time exertion and gives the attacker a considerable advance in its practices. 66, 66 % of the samples are not extensible. Thus, it is not possible to add new functionalities adapted to the learning of the malware behaviour. This possibility is only given to the tool developers It can also be considered as a way to avoid bugs and vulnerabilities in the code. It is surely because there is not a way of controlling and reviewing fairly the final software if collaboration for extensibility is possible. However, as it is open source, normally it is considered to be possibly modified by a third party under the General Public Licence (GPL) terms. The objective remains and consists of adding new modules to the tool when analysing even if afterwards the modified tool is not published. Unlike, *Radare* project [11] permits the extensibility with control process review. This accelerates the evolution and the efficiency of the software. We noticed during our evaluation that some tools integrate others (or functionalities from others) but it is not explicitly specified on the official web sites. This is a limitation for anyone interested in using them in that case, the same tasks are performed many times rather than once. For instance, *Radare* uses *soot*, and *apktool* performs very well the functionality of AXMLPrinter. On the other hand tools that use graphical user interface can be integrated into existing Java IDE (Integrated Development Environment), in order to assist the development by making the analysis comfortable.

Table2. Static evaluation

	U	D	F
Smali [9]	Command-line & shell	wiki, docs, example	Explicitly specified
Dedexer [10]	Command-line & shell	Tutorial, Blog	Explicitly specified
Radare [11]	Command line & shell	Complete and good	Explicitly specified
Dex2jar [12]	Command line & shell	Good user guide	Explicitly specified
Dare/soot [13,16]	Command line & shell	Complete tutorials	Explicitly specified
Jd-gui [14]	Easy to use interface	Pre-requisites	Explicitly specified
Jad [15]	Command line & shell	Read-me file	Explicitly specified
Apktool [17]	Command line & shell	Good installation guide, forum	Explicitly specified
AXMLprinter [18]	Command line & shell	Not documentation	Explicitly specified

	P	E
Smali [9]	Platforms with JDK	Not possible
Dedexer [10]	Platforms with JDK	Possible to extend it
Radare [11]	Platforms with JDK	Possible to extend it
Dex2jar [12]	Windows, Linux	Not possible
Dare/soot [13,16]	Linux, MAC OS[1]	Not possible
Jd-gui [14]	Linux,Windows, Mac	Integrated in Java IDE
Jad [15]	Multi-systems& multi-architecture	Not possible
Apktool [17]	Mac,windows, Linux	Not possible
AXMLprinter [18]	Platforms with JDK	Not possible

3.2 Evaluation of Dynamic Tools

Dynamic analysis refers to the execution of the mobile application in an isolated environment, such as a virtual machine or emulator, so that researchers can monitor the application behaviour dynamically. The dynamic analysis can give positive results by capturing runtime data such as system events and network. Table 3 shows the results from our data collecting. Unlike the static samples for security, dynamic tools do not provide features for interactive dialogue with the users via a UI. In this case, the user does not participate entirely in the process until the results. The tool processes the analysis in background. At the end, a report is generated. It looks pretty good without any participatory effort, but without the possibility of integrating own functionalities, the user cannot get some crucial details like explicit data flows or system calls. Once more, the user should have good skills of using command lines, in particular for administrative tasks.
For example, built for custom ROM, *TaintDroid* [24] is not quite easy to understand for a beginner. Although, there is a discussion group and an execution demo, there are several notions to understand: unlocking, loading bootloader, Flash device, and kernel. With dynamic tools, user should have goods skills in dealing with the Android platform and its components because we need to install a package to simulate its execution behaviour. Documentation must be clear for all type of users. *Systrace* and *Logcat* consist of viewing, capturing and debugging Android processes. Android developer Website [23][22] provides illustrated documentation related to them but easy for Android audience. Compared to static tools, dynamic ones are deployable in almost all systems that can host Android SDK and those that can have a web browser. Therefore, it is much easier for them to find where they are comfortable to work.
All samples of dynamic tools cannot be extended with new features.

3.3 Evaluation of integrated Environments

The other possibility to perform an analysis is to use an isolated virtual machine that integrates already dynamic and

[1] the pre-verification is not currently supported on the Mac OS version

static tools. In case of the Android platform, these tools are based on Linux. Table 4 gives their evaluation with the same criteria defined in Section 2. Most of them use both static and analysis tools studied previously. This combination gives much strength to the integrated environments (**IE**) when the user is learning capabilities of malwares. They (66,66%) are built based on Ubuntu platform rendering easy use for multiple users familiar with Ubuntu. Additionally, the usage of command line interface is primordial for the analysis.

Table 3. Dynamic evaluation

	U	D	F	P	E
Taintdroid [24]	Not easy to use for beginner. Command line.	Not easy to understand for	Explicitly specified	Platforms with Android SDK	Not possible
DroidBox	Command line	Installation steps	Not clearly specified.	Linux, Mac OS	Not possible
LogCat [22]	Interface with DDMS. Command line	Easy for Android users, not for beginners	Explicitly specified	Platforms with Android SDK	Not possible
Strace [23]	Command line	Easy for Android user, not for beginners	Explicitly specified	Platforms with Android SDK	Not possible
Mobile SandBox [21]	Simple web interface.	Clear report to user	Explicitly specified	Online	Not possible
Anubis [20]	Simple web interface	Good & complete	Explicitly specified	Online	Not possible

The use of IE supposed that static and dynamic tools are already been chosen. IE gives just a comfortable space where these tools are already installed and people can use them in Virtual machine, in an isolated manner. To apply with integrated tools, mostly command line is used even if in some cases graphical user interface exist.

(*Santoku* [25] and *Mobisec* [26]). Using IE implies the user participate entirely from (1) the installation of the Virtual Machine, through (2) the configuration to (3) the use of analysis tool The user therefore must have sufficient knowledge to use virtual machine. But once finished with the steps 1 and 2; tools can be used without too much effort. In these tools, Android SDK (Software Development Kit) is already installed. On the contrary, resources such as memory, disk space, and processor are highly required. Unlike Android Reverse Engineering, IEs provide only guide for installation's configuration. No information is pointed out for the installation of the virtual machine.

As they represent fully Linux platforms, they are extensible and the user can write scripts to adapt the learning as needed. It is possible to install additional tools and even platforms (like *DroidBox* [19], *AndroidGuard* [27]) that will aid the malware analysis.

During the investigation, the common features that come out are: reverse engineering, penetration testing, malware analysis, and device forensics. *DroidBox* and *AndroidGuard* are included in ARE [26] and *Atamer*[29]. At last, *Santoku* is the most complete IE including development tools, wireless analysers, reverse engineering, penetration testing and Device Forensics. *Androguard* is also integrated in *Santoku*. These tools are similar concerning their extensibility quality.

There are many tools used in research to scrutinize actions and behaviour of malicious programs. The analysis can be conducted inside the device or outside in a dedicated environment. However, if some qualities are not well used, they can slow down the expected effects. From our previous evaluations, rarely a tool is at the same time usable, functional, well documented, extensible and portable. This shows clearly that users must preferably focus on tools that are sufficiently documented. Otherwise, they can choose IEs that combine many tools into a system accessible through UI or command line. Since the aim is oriented to detect malicious actions, security characteristic must be analysed. The next section evaluates deeply the security aspect of these tools.

3.4 Security evaluation

We group security challenges faced during our investigation into six classes.

Update: Although one can think that for static analysis process, it is safe since malicious code actually is not running; the tool used for analysis can be vulnerable and give false results. This may also concern dynamic tools or IEs. They are subjects to updates. Or as we have seen, documentation can fail or be of poor quality. In this case the user will not be aware of the new version or existing patches to fix bugs. The lack of documentation also implies that the user is unable to practically perform the update. Another reason is the size of IEs that varies from 1GB to 5 GB. Limited with the bandwidth, the user will avoid proceeding with a new version. Moreover, documentation does not clearly specify how to use other technique such as GIT to pull updates or upgrades. IE is a combination of Virtualization machine, Linux Kernel, and integrated tools. Since the first element is written by developers, it is susceptible to contain bugs. Some of these

bugs are vulnerabilities expected by malicious to attempt to escape from Sandbox.

Sources of tools: Unfortunately, we discovered that security tools can come from non official Websites. For this reason attackers can use repackaging technique to infiltrate malicious code inside and then the user will have bad outputs unconsciously. This will lead also to data leaks from the host system. For instance, smali [9] is also found in [37] that links to [38], famous to host malicious scripts. The user will be mislead to download bad codes in its machine and will gather fake tool from attackers. The unavailability of documentation and features specified can bring one to a third webpage that is malicious.

Open access: analysis and dynamic tools used in our sample are open source. This means that attacker also can access them and use them to study rather their limitations and strengths. To confirm this point of view, it is stated in [28] that "*...the purpose is to provide **attackers** and **defenders** the ability to test their mobile environments to identify design weaknesses and vulnerabilities...*". Consequently, it is easier for attackers to adapt their code to escape analysis processes. Attackers can even redirect user to a remote server they control. Some IEs can only analyse malwares from certain size. This is an advantage for third parties.

Platform configurations: IE runs inside a virtual machine, which can be hosted in a physical system. A beginner can fail to configure the virtualization machine to be isolated from Internet and other external communication.

Table 4. Integrated environments evaluation

	U	D	F	P	E
Santoku [25]	Environment similar to Ubuntu&Interface for tools	Easy HowTo, Faq, Forum	Explicitly specified	Platforms with Virtual Machine	Many tools are integrated
ARE[2][26]	Environment similar to Ubuntu.& shell	No documentation, just screenshot	Explicitly specified	Platforms with Virtual Machine	Many tools are integrated
Mobisec[28]	Environment similar to Ubuntu.& shell, Interface for tools	Detailed installation instructions	Explicitly specified	Platforms with Virtual Machine	Many tools are integrated
Atamer[29]	Environment similar to Ubuntu & shell	Not clear installation process	Explicitly specified	Platforms with Virtual Machine	Many tools are integrated
Androguard [27]	Command line & shell	Clear documentation	Explicitly specified	Linux, Mac OS, windows	Possible to extend
ApkInspector [30]	Command line & automatic installation	Installation instructions	Explicitly specified	Linux[3]	Possible to extend modules

Consequently, if the host machine is infected, so does the integrated environment. Vice versa, the infection can come from inside (in the case for example a repackaged tool is installed and launched) and spread to outside.

Naivety of user: Other challenging security issue is the user unawareness. We have already taught about one consequence, which is to be lured in fake websites. Also the user is not expert of systems security to measure and understand sources of vulnerabilities. For instance, concerning the usability, the command line interface is more secured than the graphical one that contains many modules subject to bugs and vulnerabilities. More, even if the tool is extensible the user is not able to reinforce the security by writing new codes.

3.5. Recommendations and Discussions

Interface: Tool authors should improve the user interface by considering the security issues.

Regular updates: It is highly recommended to apply security patches as soon as it is available. Most of IEs are based on Ubuntu, so users see when available updates and should apply. We recommend to continuously monitoring the development of the tool via the corresponding website to see if there are bugs corrected or new version available.

Set a safe environment: This concerns the IEs. We recommend to use the host-only networking feature of the virtualization software and eventually to dedicate a single physical system to your virtualized environment without any network connections unless required for performing specific tasks [33].

[2] Android Reverse Engineering

[3] The latest version tested on Ubuntu 10.10 and Ubuntu 11.04.

Learning of the user: Once being on website for security tool, the user must be sensitised in not getting source code elsewhere. If the user has any doubt or question, better write directly to the author or go to the forum page. Some effort should certainly be made to have basic background knowledge on command line. This is one of the prerequisites for analysing a malware. If possible follow training on malware analysis.

Choice of tool: If the tool does not meet certain requirements, find another one.

Documentation: The documentation should be updated according to the improvements of the tool. It should be available to clearly guide the user. Often, video tutorial are more expressive and helpful for new user than the text one.

Security aspects can be respected only if the user is aware and meticulous. However, it is not a guarantee that the good security education of the user will block attacks. Google Play is always subject to attacks even if review and control are performed for each submitted app. Basic measures like avoid downloading a tool from unofficial website should be respected.

4. CONCLUSION

Researchers use reverse engineering, dynamic and integrated environment to learn from malware apps in order to block them. Our study shows that they lack quality characteristics. We rarely found a tool with concise and understandable documentation, clearly specified features, easy to use, extensible, and portable at the same time. This tends to hinder users from using tools to perform analysis giving, therefore, advantage to the attackers. Our evaluation shows that these tools can be trigger elements for malwares inside the malware analysis platforms. The user will therefore be victim of attacks unconsciously, while downloading a tool from a fake website. However, to mitigate these problems the user should be aware of security issues. Tool authors can rely on their website to sensitise the users. Furthermore, the users should be concise in their choice of tool according to features and quality characteristics.

As a future work, we plan to conduct experimental analysis with different users to determine their awareness on security tools.

REFERENCES

[1]. Felt, A.P., Finifter, M., Chin, E., Hanna, S. and D. Wagner, "A Survey of Mobile Malware in the Wild," Proc. ACM Workshop Security and Privacy in Mobile Devices (SPMD 11), ACM, 2011, pp. 3-14.

[2]. Enck, W., "TaintDroid: An Information-Flow Tracking System for Real-Time Privacy Monitoring on Smartphones," Proc. 9th Usenix Symp. Operating Systems Design and Implementation (OSDI 10), Usenix, 2010;

[3]. Zhou, Y. and Jiang, X., "Dissecting Android Malware: Characterization and Evolution," Security and Privacy (SP), 2012 IEEE Symposium on, vol., no., pp.95, 109, 20-23 May 2012

[4]. Milbourne, G. and Orozco, A. Android Malware Exposed: An In-depth Look at the Evolution of Android Malware, Virus Bulletin, August 2012

[5]. Castillo, C.-A, Android Malware - Past, Present, and Future. McAfee, Inc. 2011

[6]. Google's Android becomes the world's leading smart phone platform, http://www.canalys.com/newsroom/google%E2%80%99s-android-becomes-world%E2%80%99s-leading-smart-phone-platform, accessed on 07.06.2013

[7]. Lookout Mobile Threat Report, August 2011, https://www.lookout.com/_downloads/lookout-mobile-threat-report-2011.pdf, accessed on 07.06.2013

[8]. SecMobi Wiki, a collection of mobile security resources, http://wiki.secmobi.com, accessed on 07.06.2013

[9]. Smali, http://code.google.com/p/smali/, accessed on 07.06.2013

[10]. Dedexer, http://dedexer.sourceforge.net/, accessed on 07.06.2013

[11]. Radare, http://radare.org, accessed on 07.06.2013

[12]. Dex2jar, http://code.google.com/p/dex2jar/, accessed on 07.06.2013

[13]. Dare, http://siis.cse.psu.edu/dare/index.html, accessed on 07.06.2013

[14]. JD-GUI, http://java.decompiler.free.fr/?q=jdgui, accessed on 07.06.2013

[15]. JAD, http://www.varaneckas.com/jad/, accessed on 07.06.2013

[16]. Soot, http://www.sable.mcgill.ca/soot/, accessed on 07.06.2013

[17]. Android-apktool, a tool for reverse engineering Android apk files, http://code.google.com/p/android-apktool/, accessed on 07.0621013

[18]. android4me, J2ME port of Google's AndroidB http://code.google.com/p/android4me/, accessed on 07.06.2013

[19]. Droidbox, Android Application Sandbox http://code.google.com/p/droidbox/, accessed on 07.06.2013

[20]. Anubis: Analyzing Unknown Binaries, http://anubis.iseclab.org/, accessed on 07.06.2013

[21]. Mobile Sandbox http://mobilesandbox.org/, accessed on 07.06.2013

[22]. Logcat, http://developer.android.com/tools/help/logcat.html, accessed on 07.06.2013

[23]. Systrace, http://developer.android.com/tools/help/systrace.html, accessed on 07.06.2013

[24]. Realtime Privacy Monitoring on Smartphones, http://appanalysis.org/, accessed on 07.06.2013

[25]. Santoku, https://santoku-linux.com/, accessed on 07.06.2013

[26]. ARE, https://redmine.honeynet.org/projects/are/wiki, accessed on 07.06.2013

[27]. Androguard, http://code.google.com/p/androguard/, accessed on 07.06.2013

[28]. Mobisec, http://sourceforge.net/projects/mobisec/, accessed on 07.06.2013

[29]. Android Tamer, http://androidtamer.com/, accessed on 07.06.2013

[30]. apkinspector, http://code.google.com/p/apkinspector/, accessed on 07.06.2013

[31]. Rubin, A. Google+ post on the Android ecosystem https://plus.google.com/u/0/112599748506977857728/posts/Btey7rJBaLF, accessed on 07.06.2013

[32]. Felt, A.-P., Greenwood, K. and D. Wagner, "The Effectiveness of Application Permissions," Proc. 2nd Usenix Conf. Web Application Development (WebApps 11), Usenix 2011

[33]. Sirorski, M., Honig, A.Practical Malware Analysis, The Hands-On Guide to Dissecting Malicious Software, no starch press, San Francisco, 2012

[34]. Android Reversing analysis, http://wiki.secmobi.com/tools:android_reversing_analysis, accessed on 07.06.2013

[35]. Android Dynamic analysis http://wiki.secmobi.com/tools:android_dynamic_analysis, accessed on 07.06.2013

[36]. Android Integrated Analysis, http://wiki.secmobi.com/tools:android_integrated_analysis_environment, accessed on 07.06.2013

[37]. Baksmali / Smali Android's dex files (using command line tool), http://www.wrapcode.com/android/baksmali-smali-dex-files/); accessed on 07.06.2013

[38]. MediaFire Smali link, http://www.mediafire.com/download/s1cb90m8cf9x9hi/Smali-Me-1.0.zip, accessed on 07.06.2013