ANDROID FORENSIC AND ANTI-FORENSIC TECHNIQUES – A SURVEY

NEMANJA D. MAČEK
School of Electrical and Computer Engineering of Applied Studies, Belgrade and SECIT Security Consulting, nmacek@viser.edu.rs

PERICA ŠTRBAC
School of Electrical and Computer Engineering of Applied Studies, Belgrade, pericas@viser.edu.rs

DUŠAN ČOKO
School of Electrical and Computer Engineering of Applied Studies, Belgrade, dusanc@viser.edu.rs

IGOR FRANC
Belgrade Metropolitan University, Faculty of Information Technologies and SECIT Security Consulting, igor.franc@metropolitan.ac.rs

MITKO BOGDANOSKI
Military Academy General Mihailo Apostolski, Skoplje, Macedonia, mitko.bogdanoski@ugd.edu.mk

Abstract: *Technology concerning mobile devices has presented revolutionary growth during the last decade. Mobile phones do not serve only as a means of communication, but also as portable computers with advanced communication capabilities. Smartphones are able to store a rich set of personal information and at the same time provide powerful services, such as location-based services, Internet sharing via tethering, and intelligent voice, thus increasing the likelihood of a such devices being involved in a criminal activities. Mobile forensics is the science of recovering digital evidence from a mobile device under forensically sound conditions using accepted methods. During the last few years, a significant amount of research has been conducted, concerning various mobile device platforms forensics, data acquisition schemes, and information extraction methods. This paper provides a comprehensive overview of the field, by presenting a detailed assessment of methodologies regarding Android forensic and anti-forensic techniques.*

Keywords: *Android, Phone, Forensics, Anti-forensics*

1. INTRODUCTION

The Android mobile platform has quickly risen from its first phone in October 2008 to the most popular mobile operating system in the world by early 2011. According to Gartner, Inc., global sales of smartphones to end users totaled 349 million units in the first quarter of 2016, with Android Android regaining share over iOS and Windows to achieve 84 percent share [1]. The explosive growth of the platform has been a significant win for consumers with respect to competition and features. However, forensic analysts and security engineers have struggled as there is a lack of knowledge and supported tools for investigating these devices [2]. Criminals could use Android phones for a number of activities ranging from harassment through text messages and e-mail frauds to trafficking of child pornography and communications related to narcotics. The data stored on these phones could be extremely useful to analysts through the course of an investigation of these activities. Unless anti-forensics is somehow deployed, a large volume of probative information linked to an individual exists on every Android phone, including call history, contacts, messaging data, e-mails, browser history and chat logs. According to Lessard and Kessler, these phones have more probative information that can be linked to an individual per byte examined than most computers [3]. However, this data is harder to acquire in a forensically proper fashion due to a wide range of phones available and a general lack of hardware and software standardization. As an example, even different models of the same manufacturee sometimes require different data cables and software to access the phone via computer.

Roughly, one may distinguish three types of scenarios where Andoid forensics may come in handy: an investigation that will adjudicated in a criminal or civil court of law, internal corporate investigations (intellectual property, data theft, inappropriate use of company resources or employment related investigations) and investigations that target family matters (divorce, child custody or estate disputes).

Having that said, one may ask a question: where does the anti-forensics fit in? Majority of users do not employ adequate security meassures on their Android phones. So, let's observe the following scenario: a user that is not involved in anything related to crime does not employ a pattern to unlock the screen. The very same user does not have anti-theft software installed but somehow manages to loose his phone. A malicious person that has found the lost Android phone now has a temporary access to Gmail, Facebook, Twitter and all other accounrs that the user was logged in to. Authors will allow readers to conclude the story by themselves (suggestion: avoid happy endings). According to the aforementioned scenario, the data stored on a phone presents an obvious threat to the user's privacy. Data also provides a well-defined profile of the user that can further be used to reconstruct his actions at a specific time [4]. A user who wants to protect his privacy can employ anti-forensics techniques. According to Ryan Harris, "anti-forensics is considered to be any attempt to compromise the availability or usefulness of evidence to the forensics process. Compromising evidence availability includes any attempts to prevent evidence from existing, hiding existing evidence or otherwise manipulating evidence to ensure that it is no longer within reach of the investigator. Usefulness maybe compromised by obliterating the evidence itself or by destroying its integrity" [5].

This paper briefly analyses some of the Android forensic and anti-forensic techniques reported in the literature. The rest of the paper is organized as follows. Section 2 briefly describes Android operating system, certain security issues and discusses the features common to all devices that are fundamental to forensic investigation. A survey of forensic solutions reported in the literature and anti-forensic techniques is given in sections 3 and 4, respectively and section 5 concludes.

2. ANDROID OPERATING SYSTEM

Android, a mobile operating system developed by Google, is the best-seller for tablets since year 2013, and on smartphones it is dominant by any metric [6]. The middleware, libraries and APIs written in C and software running on an application framework which includes Java-compatible libraries reside on top of the Linux kernel (see Image 1).

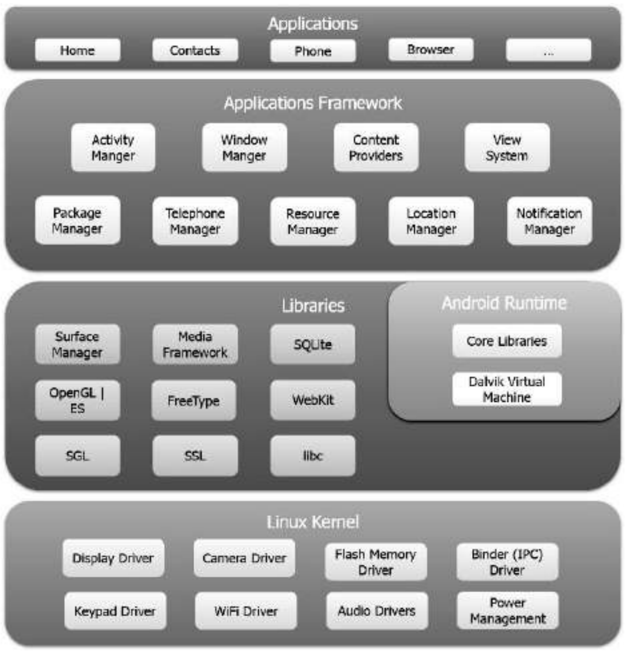

Image 1: Android software stack

Linux kernel is developed independently of other Android's source code and provides the support for some fundamental functions, such as device drivers, network infrastructure and power management [7, 8]. Libraries and Android runtime reside in the next level of the architecture. Libraries provide the infrastructure for applications to run properly, such as binaries and graphics support. Android's runtime consists of the Dalvik Virtual Machine (DVM) and the core libraries that provide the available functionality for the applications [7]. Its main purpose is the creation of a stable and secure environment in which applications are executed: each application runs in its own sandbox and therefore is not affected by other applications or system functions. A satisfying level of security is preserved by allowing certain resources to be used only if permitted by special privileges. The rest of the architecture consists of the applications framework and the applications layer that manage general application structure, such as containers, alerts and the applications themselves. As Android runtime libraries are written in Java, DVM translates Java to a language that the OS can perceive [9] – until version 5.0, Android used Dalvik as a process virtual machine with trace-based just-in-time compilation to run Dalvik executable code, which is usually translated from the Java bytecode. Following the trace-based just-in-time principle, in addition to interpreting the majority of application code, Dalvik performs the compilation and native execution of select frequently executed code segments each time an application is launched [10, 11].

Due to the small chip size, non-volatile nature and energy efficiency, NAND flash memory was selected to serve as Android storage [12]. Yet Another Flash File System 2 (YAFFS2) was the first filesystem implemented on devices running Android, but, due to certain limitations (such as large file coverage) [13], was replaced with Ext4 before the release of Android version 2.3 (Gingerbread). The Ext4 filesystem, apart from successfully coping with the weak points of YAFFS2, is enhanced with the journaling event function which provides recovery options and facilitates acquisition of unallocated files [13, 14]. As NAND flash memory was incompatible to the Linux kernel, a new technique was implemented to provide the ability to access the flash memory areas [8]. The Memory Technology Devices (MTD) system was one of the facilities serving as an intermediary between the kernel and the file system and is present in many Android devices. Handsets that do not support the MTD system usually utilize the plain Flash Transaction Layer (FTL) that enables communication between the two parts [14]. The flash storage on is split into several partitions: operating system resides on /system while /data is used to store user data and application installations. As root access is not gained to users /system and sensitive partitions are mounted read-only, unless device is rooted by exploiting security flaws.

Security and privacy issues of Android devices can be classified either as issues arising from surveillance by public institutions, such as NSA (see [15, 16] for more details), common security threats, such as malware that sends text messages from infected phones to premium-rate telephone numbers without the consent or even knowledge of the user [17] or technical security features, typically resulting from unnecessary permissions required to install applications. As stated before, applications run in a sandbox, unless access permissions are explicitly granted by the user when the application is installed. This reduces the impact of vulnerabilities and bugs in applications, but the unnecessary required permissions that result from either developer confusion or lack of documentation work against effectiveness of sandboxing. Although since the version 6.0, users are allowed to block applications from having access to the contacts, calendar, phone, sensors, SMS, location, microphone and camera [18], full permission control is only possible if device is rooted.

So which features are common to any Android device and can they be used in the forensic investigation? According to Andrew Hoog [2], Android was engineered from the beginning to be online, whether using cellular or wireless networks. Being online is a prerequisite that allows the execution of another fundamental feature: downloading and installing applications from the Play Store. To a user, this feature presents the ability to extend the functionality of the device. To a forensic investigator, applications downloaded from the Store present a rich source of information. Finally, the ability for users to store their data on the devices is important as much to a forensic investigator as it is important to a user himself. Typically, stored data is the basis behind any forensic investigation.

3. ANDROID FORENSICS: A BRIEF SURVEY

Procedure for handling Android devices contains several steps, such as securing the device, isolating it from the network, circumventing the pass code and imaging mass storage devices. Depending on the way how data is accessed, android forensic techniques can be classified either as logical or physical. Logical technique extracts allocated data, typically by accessing the filesystem, with the exception of SQLite database (that might still contain deleted records in the database). Physical techniques, on the other hand, extract data from the physical storage medium directly and do not rely on the filesystem. There are advantages to this approach and the most significant is that with the physical forensic techniques it is possible to recover both the allocated and the unallocated (deleted or obsolete) data. One of the guiding principles of any forensic investigation is to avoid modification of the target device in any manner, and this principle works for Android devices also. The rest of this section will provide a review of the forensic techniques, solutions and methods reported in the recently published literature of interest.

Lai et al. [19] implemented a live-forensic acquisition procedure, based on commercial forensic suites through cloud computing, designed for Android devices. Although acquisition type was not specified, the procedure resembled to logical acquisition that can be applied to rooted devices as well. Since proper time-stamping is an essential for the validity and integrity of forensic evidence, actual date correction is another interesting feature in their approach.

Simao et al. [9] proposed a forensic acquisition framework for the Android in the form of flowchart, applicable to many scenarios, including damaged devices and fragmented memory page analysis. In order to validate the model, authors have conducted experiments on devices with different conditions and figured out that the proposed scheme was applicable. Downside of their solution is lack of some crucial elements necessary for real-time investigation.

Research of Vidas et al. [8] deals with the forensic acquisition on devices protected by a screen lock. Since a brute-force attack on the device could lead to a further block, and possibly to inevitable data modification, another technique had to be implemented. To resolve the problem, authors have stated that booting with a recovery image could easily bypass any kind of active lock code. Recovery mode boot file residing in the Android root was significant for the acquisition process of the recovery image, as by booting into recovery mode, the boot process is circumvented with the boot target set to boot image currently loaded in the recovery partition. Boot image that authors have used consists of existing modified files and variety of transfer daemons and binaries. The authors have noticed that boot options differ between brands of mobile phones and have examined several different case studies. Downside of aforementioned research is the lack of statistic results of data retrieval.

Sylve et al. [20] referred to a lack of studies applicable to physical acquisition and highlighted the importance of this issue. The researchers presented "a methodology for acquiring complete memory captures from Android, code to analyse kernel data structures and scripts that allow analysis of a number of user and filesystem based activities". Authors have also enumerated the existing methodologies on volatile memory analysis for Linux and Android operating systems and compared the capabilities of the corresponding tools. The results of their experiments provided a proof that Linux oriented forensic techniques were not compatible to the Android.

Andriotis et al. [21] implemented a forensic acquisition method that employs WiFi and Bluetooth. The most significant parts of their research was the fact that devices used were involved in actual crime scenes. Afterward, they presented a detailed step-by-step procedure to complete logical acquisition, which was common for all the devices participating in the experiment, which was considered a success as any critical evidence was recovered in every networking attribute.

Ext4 filesystem that became the successor of YAFFS was examined by Kim et al. [13]. Authors have used two rooted devices running Android and their research was limited to logical acquisition. Detailed description of the file system was provided and forensic acquisition for the journal log area was summarized.

Mylonas et al. [22] studied the involvement of context-measuring devices, such as accelerometers and GPS, in mobile forensics. Authors have stated that this kind of data can be of great importance and that a special approach is required because of the volatile nature of the data itself. Methodology on data acquisition from sensors was proposed, ranging from theory to practical procedures executed at the laboratory level. Data acquisition system they have developed took into consideration security mechanisms on the target devices as well as the procedures to bypass these mechanisms. As the system they have developed consists of two parts (the workstation and the mobile agent), and as one of the possible use of the solution would be to acquire data from a phone belonging to a potential suspect, agent installation had to be forced and functionality to be obfuscated either via social engineering or fake error messages. According to their research, 12 out of 15 sensors need absolutely no permission to gain access to, leading to conclusion that security behind sensors is easy to bypass: agent is triggered each time the user accesses a sensor, acquires the data, encrypts it and sends it to a workstation if the device is connected to a network.

Live forensic methods as a means for surveillance of malware activity on Android is presented in work of Guido et al. [23]. The solution was developed as a mainstream Android application in order to avoid rooting. It comprised of five modules programmed in Python, each detecting changes in specific parts of the operating system: bootloader, recovery, filesystem, deleted files and APKs. Experiments consisted of three rounds of malware injections on target mobile devices, with many successful detections, but weak points, such as false positives and inability to detect some deleted entries have occurred also. Despite the defects, the solution proposed in the paper was one of the important contributions to the Android forensics. Similarly, Justin Grover [24] has developed DroidWatch application that performs continuous tracking of events and data flow on an Android device and sends the information to a Web Server. As rooting of the device was avoided due to authors policy, the range of acquired data was limited. The data process flow within the DroidWatch app is depicted in Image 2. Data collection and storage is a continuous process, with transfers scheduled by configurable variable. Upon a successful transfer to the enterprise server, events dated prior to the transfer are wiped from the local phone database. File transfer attempts that fail are logged in the database and do not result in the wiping of any events.

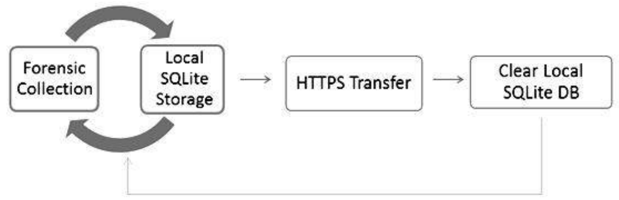

Image 2: DroidWatch data process flow diagram [24]

Son et al. [25] conducted an evaluation on the Recovery Mode method with seven rooted Samsung devices taking part as a sample. The results from the use of JTAG method served as a comparison vector to the Recovery Mode. A section was dedicated on the acceptable practices during the data acquisition phase in Recovery Mode. A flowchart related to the steps taken during the acquisition procedure was introduced, the importance of using the appropriate bootloader for each device was pointed to and issues with encrypted ones were mentioned. Actions that should have been taken into consideration during the restoration process have been highlighted, for example the prohibition of interaction with the menu elements in Recovery Mode and the USB cable separation from the device before battery removal. Additionally, custom software was developed in order to conduct the data extraction tasks and check the integrity of the method. Finally, the hash values of the data partition that were extracted in both cases was calculated and proved to be equal, assuming that integrity was preserved.

Muller and Spreitzenbarth have investigated innovative techniques in an effort to assess how much valuable information can be extracted from encrypted Android phones [26]. A cold boot attack was performed by freezing to gain the device in order physical access to the device memory and acquire information such as encryption keys or personal data. The method, however, has an important limitation: the user partition gets wiped out when the device bootloader unlocks. Still, it is the first work to perform a successful and effective cold boot attack on Android phones and the implementation of cryptographic solutions does not appears as a problem that cannot be bypassed.

Konstantia Barmpatsalou et al. provided a comprehensive review of forensic techniques applicable to other smartphone operating systems [14].

4. ANDROID ANTI-FORENSIC TECHNIQUES

As stated before, the purpose of anti-forensics is to compromise the availability or usefulness of forensic evidence. Distefano et al. distinguished several kinds of anti-forensics [27]: destroying evidence (making it unusable during the investigation), hiding evidence (subverting an analyst by decreasing the visibility of the evidence), eliminating sources (neutralization of the evidentiary sources) and counterfeiting evidence (creation of a fake version of the evidence which is properly made to carry wrong or deviated information in order to divert the forensic process).

Kessler [28] categorises anti-forensics into four groups: data hiding, artefact wiping, trail obfuscation, and attacks against forensics processes or tools, which refer to attacks that force the forensic analyst to perform non-standard procedures or call into question the data recovered. For computer anti-forensics, data hiding contains things like steganography, deleted files, and storing data in the cloud or in other storage space. On a non-rooted phone, information can be hidden by having an application store it somewhere secluded and restore it at a later time. This approach also allows quick mass-deletion [27]. Artefact wiping refers to overwriting data down to the level where it is impossible to restore it from, even with high-tech un-deletion techniques. Two weaknesses with this class however may be noticed: they may miss some data, and they may leave traces of the wiping that have occurred (probably the wiping tool itself will remain). Since Android anti-forensics is mainly concerned with data legitimately stored and usable on the phone, and not with attacks or traces on other devices on the network, trail obfuscation is not considered to be very relevant anti-forensic technique. Trail obfuscation typically refers to network forensics. When an attacker does not need a reply, he can spoof the sender's address to make tracing the attack to its source harder. It is also possible to use spoofed sender addresses for attack amplification, by tricking third parties into sending much more traffic to a victim than the attacker could on their own. Other tools in this category, such as onion routers, Web proxies and e-mail anonymisers, hide the real sender of traffic behind a server which serves many clients. Trail obfuscation also includes log file and timestamp alteration.

5. CONCLUSION

Variety of conducted research on Android, and in general, mobile forensics, as well as undergoing standardization attempts indicate that the area is under continuous development. The work presented in this paper provided a comprehensive review of the state-of-the-art research in the field of Android forensics, as well as a classification of important Android anti-forensic techniques. Any relevant current work, be it a research or review, can be used as a reference to anyone interested in better understanding the facts of this rapidly evolving and interesting research discipline.

REFERENCES

[1] Gartner Inc., "Gartner Says Worldwide Smartphone Sales Grew 3.9 Percent in First Quarter of 2016", Egham, UK, May 19, 2016.

[2] A. Hoog, "Android forensics: investigation, analysis and mobile security for Google Android", Elsevier, 2011.

[3] J. Lessard, G. Kessler, "Android Forensics: Simplifying Cell Phone Examinations", Small Scale Digital Device Forensics Journal, 4(1), pp. 1-12, 2010.

[4] P. Albano, A. Castiglione, G. Cattaneo, A. De Santis, "A Novel Anti-Forensics Technique for the Android OS", 2011 International Conference on Broadband and Wireless Computing, Communication and Applications, pp. 380-385, 2011, IEEE.

[5] R. Harris, "Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem". Digital Investigation 35, pp. S44-S49, 2006.

[6] F. Manjoo, "A Murky Road Ahead for Android, Despite Market Dominance", the New York Times, May 27, 2015, ISSN 0362-4331.

[7] I. I. Yates, "Practical investigations of digital forensics tools for mobile devices. In 2010 Information Security Curriculum Development Conference, October 2010, pp. 156-162, ACM.

[8] T. Vidas, C. Zhang, N. Christin, N., "Toward a general collection methodology for Android devices", Digital investigation 8, pp. S14-S24, 2011.

[9] A. Simao A, F. Sicoli, L. Melo. F Deus, JR Sousa, "Acquisition and analysis of digital evidence in android smartphones", International Journal of Forensic Computer Science, Vol. 6, No. 1, pp. 28–43, 2011.

[10] B. Cheng, B. Buzbee, "A JIT Compiler for Android's Dalvik VM" (PDF), android-app-developer.co.uk, Google, pp. 5–14, May 2010.

[11] H. Q. Raja, "Android Partitions Explained: boot, system, recovery, data, cache & misc", Addictivetips.com., May 19, 2011.

[12] C. Zimmermann, M. Spreitzenbarth, S. Schmitt, FC. Freiling, "Forensic analysis of yaffs2", In Sicherheit, pp. 59–69, 2012.

[13] D. Kim, J. Park, K-g Lee, S. Lee S, "Forensic analysis of android phone using ext4 file system journal log", in Future Information Technology, Application, and Service, Springer Netherlands, pp. 435-446, 2012.

[14] K. Barmpatsalou, D. Damopoulos, G. Kambourakis, V. Katos, "A critical review of 7 years of Mobile Device Forensics", Digital Investigation, 10(4), pp. 323-349, 2013.

[15] Staff, "Privacy Scandal: NSA Can Spy on Smart Phone Data", September 7, 2013.

[16] James Ball, "Angry Birds and 'leaky' phone apps targeted by NSA and GCHQ for user data", theguardian.com, Jaunary 27, 2014.

[17] E. Protalinski, "Android malware numbers explode to 25,000 in June 2012". ZDNet, July 17, 2012.

[18] R. Amadeo, "Android 6.0 Marshmallow, thoroughly reviewed", Ars Technica, May 10, 2015.

[19] Y. Lai, C. Yang, C. Lin, T. Ahn, "Design and implementation of mobile forensic tool for android smart phone through cloud computing". In International Conference on Hybrid Information Technology, pp. 196-203. Springer Berlin Heidelberg, 2011.

[20] J. Sylve, A. Case, L. Marziale, GG. Richard, "Acquisition and analysis of volatile memory from android devices". Digital Investigation 8, No. 3, pp. 175-184, 2012.

[21] P. Andriotis, G. Oikonomou, T. Tryfonas, "Forensic analysis of wireless networking evidence of android smartphones", In 2012 IEEE International Workshop on Information Forensics and Security (WIFS), pp. 109-114. IEEE, 2012.

[22] A. Mylonas, V. Meletiadis, L. Mitrou, D. Gritzalis, "Smartphone sensor data as digital evidence", Computers & Security 38, pp. 51-75, October 2013.

[23] M. Guid, J. Ondricek, J. Grover, D. Wilburn, T. Nguyen, A. Hunt, "Automated identification of installed malicious android applications", Digital Investigation 10, pp. S96-S104, 2013.

[24] J. Grover, "Android forensics: automated data collection and reporting from a mobile device", Digital Investigation 10, pp. S12-S20, 2013.

[25] N. Son, Y. Lee, D. Kim, J.I, James, S. Lee, K. Lee, "A study of user data integrity during acquisition of android devices", Digital Investigation 10, pp. S3-S11, 2013.

[26] T. Muller, M., Spreitzenbarth, "Frost", In: Jacobson M, Locasto M, Mohassel P, Safavi-Naini R, editors., Applied cryptography and network security, Lecture notes in computer science, vol. 7954. Berlin, Heidelberg: Springer, pp. 373–88, 2013.

[27] A. Distefano, G. Mea, F. Pace, “Android anti-forensics through a local paradigm”, Digital Investigation 7, pp. S83-S94, 2010.

[28] G. Kessler, “Anti-forensics and the digital investigator”, In Proceedings of the 5th Australian digital forensics conference, December 2007.