A Novel Anti-Forensics Technique for the

Android OS

Pietro Albano∗, Aniello Castiglione†, Giuseppe Cattaneo§, Alfredo De Santis‡

Dipartimento di Informatica “R.M. Capocelli” Universita` degli Studi di Salerno

I-84084 Fisciano (SA), Italy

[pietro.albano@gmail.com∗](mailto:pietro.albano@gmail.com), [castiglione@ieee.org†](mailto:castiglione@ieee.org), [cattaneo@dia.unisa.it§](mailto:cattaneo@dia.unisa.it), [ads@dia.unisa.it‡](mailto:ads@dia.unisa.it)

Abstract—In recent years traditional mobile-phones, used only to make calls and send text messages, have evolved into even more versatile and powerful devices (SmartPhone, Tablet, etc.). These devices use a NAND flash memory type to store data, due to it being a memory that has been optimized for the fast updating of data. These flash memory drives contain sensitive data that could be a possible danger to the user’s privacy.

This paper proposes a new Anti-Forensics technique for mobile devices with the Android OS. The technique makes it possible to modify and erase, safely and selectively, the digital evidence on an Android device without having to use any cryptographic primitives or make any file system changes. While the use of cryptographic primitives or changes to the file system create con- siderable suspicion in a forensic analysis, the proposed technique uses simple software tools commonly used in \*nix-like OSes such as Android.

Index Terms—Digital Forensics; Mobile Forensics; Anti- Forensics; Mobile Anti-Forensics; Counter-Forensics; Android OS; Android Forensics; Android Anti-Forensics; Flash Memory; NAND; Secure Deletion; Sanitization.

I. INTRODUCTION

Gartner claimed [1] that in 2011 468 million mobile devices would be sold, i.e. 57.7% more than last year. In addition, according to the same research agency, in 2012 there will be 632 million smartphones in the world and in 2015, 1.1 billion. In 2012, half (49.2%) of the 632 million smartphones in circulation will work on the Android OS.

The data stored in a smartphone can be a threat to the user’s privacy. It is worth recalling that a classic mobile phone contains a large amount of private information (contacts, text messages, call lists). Additional sensitive information stored on smartphones includes e-mails, websites visited, chat messages, geo-location information, etc.. All this information can give a well-defined profile of its user in order to reconstruct his actions at a specific time. An individual who wants to protect his privacy and in particular delete or modify in a safe and selective way any digital evidence stored on the smartphone can use Anti-Forensics techniques.

A definition of Anti-Forensics, according to [2], is “...we will consider anti-forensics to be any attempts to compromise the availability or usefulness of evidence to the forensics

Corresponding author: Aniello Castiglione, Member, IEEE, [cas- tiglione@ieee.org, Phone:](mailto:castiglione@ieee.org) +39089969594, FAX: +39089969821

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.”.

While there are currently no papers dealing with the prob- lem of secure modification in flash memories (to the best of the authors’ knowledge), several studies have recently proposed techniques for the secure erasing of NAND flash memory drives that are commonly used in smartphones.

Flash memory drives (NANDs) differ from hard drives in both the technology used to store data (flash chips vs. magnetic disks) as well as the algorithms used to manage and access the data. NANDs maintain a layer of indirection between the logical block addresses that computer systems use to access data and the raw flash addresses that identify physical storage. The layer of indirection enhances NAND performance and reliability by hiding the idiosyncratic interface of the flash memory and managing its limited lifetime. However, it can also produce copies of the data that are invisible to the user but that a smart attacker can recover. The differences between NANDs and hard drives make it uncertain whether techniques and commands developed for hard drives will be effective on NANDs.

In [3] and [4], all the data is encrypted, each file has its own encryption key which is stored in the header of the file. When wanting to delete a single file, simply delete or overwrite the header. Encrypting the file system or modifying it in order to apply anti-forensics techniques creates a great deal of suspicion during a forensic analysis.

In [5], exploiting a security feature of Android, it is possible that digital evidence is hidden in a private folder which is inaccessible to third-party applications. The private folder is a private directory, created when an application is installed, in which it is possible to save any file type (e.g., text files, multimedia files). According to [5], when a given common Android application is uninstalled, the entire set of the related information, including data files and directories, is logically deleted from the file system.

This paper introduces a new anti-forensics technique for mobile devices running the Android OS that makes it possible

to edit and delete, securely and selectively, the digital evidence generated by the Android OS by using simple software tools, without making changes to the file system and using any cryptographic primitives.

The paper is organized as follows: Section II briefly de- scribes the Android OS, paying particular attention to the type of file system generally adopted by Android devices as well as the media storage (NAND) generally used. Section III contains the main forensic techniques used on Android devices. Sec- tion IV describes the proposed anti-forensics technique, while Section V describes the examples, on Android devices, of the technique that have been designed, implemented and tested. The conclusions are made in Section VI.

II. THE ANDROID OS

Android is a set of open-source software elements specif- ically designed and developed by Google for mobile devices. Although it has been designed and developed for mobile de- vices (e.g., smartphones, tablets, etc.), it includes the Operating System, a middleware and a set of applications.

A. The Android File System

The vast majority of mobile devices running the Android OS uses, and supports, the YAFFS (Yet Another Flash File System) file system, developed in 2002 and specifically to be used on NAND flash memories. It is also completely open-source. To the authors’ knowledge, YAFFS is the only file system type which fully supports flash memory technology [6].

YAFFS has several interesting features such as journaling, error correction and verification techniques, which are very useful when dealing with hardware failures/problems that are intrinsic to the flash memory technology. The Android OS is based on the Linux kernel v2.6 and, thanks to it, allows for the distribution of its file system on different storage devices, on both the internal flash memory as well as a microSD that can be inserted into the mobile device. Table I describes the structure and organization of the file system present on the mobile device adopted for the case study. The organization of the partitions as well as the mount points shown in I may change according to different devices. In the column named “Partition”, the storage device and the partitions of the internal flash memory as well as the microSD are listed. The partitions host different data: mtd0 contains miscellaneous data, mtd1 the recovery image, mtd2 the boot partition, mtd3 the system files, mtd4 the system cache and mtd5 the user data. The “Mount Point” column lists only those partitions which are accessible from the file system, in other words, only the partitions that can be mounted on a branch of the file system. The last column of Table I shows whether the partition is present on the “Internal” flash memory or the “External” microSD. The most interesting partitions to be considered during a digital forensics analysis are the mtd3 and mtd5 which contain, respectively, the core of the OS and the work space of the user.

The partition mmcblk0p1, usually formatted with the FAT

file system type, represents the external memory of the mobile

device, with it being managed by the user. The partition mmcblk0p2, which is formatted with the EXT3/EXT4 file system type, starting from Android v2.2 is used as an extension of the internal memory of the mobile device. This partition can only be accessed by the OS and the root user.

B. NAND Flash Memory

Flash memory is a solid-state storage technology and thus non-volatile, which can be used as a read-write memory due to its performance. There are principally two kinds of flash memory: NOR and NAND. These two types of memory differ in their architecture as well as in the way they are programmed. In addition, there is a hybrid flash memory, AND flash, which takes advantages of the characteristics of both NOR and NAND. NAND flash memories are optimized for the fast updating of data. It is also worth noting that the smallest block that can be deleted in a NAND is 8 Kb while in a NOR, it is

64Kb.

When data in a flash memory are updated, it is not possible to program the same page due to the one-way programming feature of flash technology, thus the page containing the to- be-updated data is entirely rewritten to a new location either in the same block or not. In the spare area, the page with new data is marked as allocated, while the old one is marked as unallocated (deleted or obsolete) [7].

C. Digital Evidence in Android

The partition mtd5 maintains the third party applications as well as all the logs related to these applications or generated by the OS itself. In [8], some of the most useful tools in a digital forensics investigation on an Android OS device are presented and analyzed.

III. ANDROID FORENSICS

In this section, the main tools for the forensic acquisition and analysis of devices equipped with the Android OS are de- scribed. Forensic acquisitions can be performed using logical and/or physical operations. Logical operations are performed only on allocated data and are usually executed by accessing the file system. Allocated data is organized in the file system structure and is accessible through it. Physical operations, on the other hand, do not use the file system to access the data but interact directly with the physical storage media.

A. Tools for the Logical Acquisition

There are essentially three different kinds of logical acquisi- tion tool: Nandroid Backup is a set of tools and a script which make it possible to backup an Android device provided that the root privileges have been granted. The backup can be considered as a logical copy of the flash memories involved (the whole internal memory and /sd-ext of the microSD). Third party applications are those developed by the developer community using the standard Application Programming Interfaces (API) supplied by Google. These APIs can access the entire device including digital evidence present on the mtd5 partition. There are several applications capable of extracting useful logical

information from the file system. These include the AFLogical

Tool developed by ViaForensics [9].

There are a few commercially available logical acquisition tools, such as Oxygen Forensics Suite 2011 by Oxygen, An- droid Logical Plug-in by Paraben and Mobile Phone Examiner by Access data.

B. Tools for the Physical Acquisition

Android is a Linux-based OS and makes the same tools present on all the Linux boxes available. In particular, the command dd makes a bit-level copy of both the internal flash memory as well as the microSD. The copy is stored on a partition located on the microSD. In order to use the dd command properly, so as to make a physical copy, it is necessary to open a remote shell on the device by using a PC. It is important to avoid any modification to the files present on the device to be acquired. The Android Debugger tool (ADB) makes it possible to remotely connect to the device and perform a bit-level copy while the device is in recovery mode.

For example, a bit-level copy of the partition

/dev/mtd/mtd5 can be acquired, with its image being stored on the microSD, by using the command:

# dd if=/dev/mtd/mtd5 of=/sdcard/userdata.dd bs=4096

C. Tools for the Analysis

The forensic, logical and physical copies of the device can be analyzed by using commercial and open-source tools, that were originally designed for PCs, such as Encase, FTK, Oxygen, Autopsy, PhotoREC. A useful open-source tool is Unyaffs [10] which extracts the logical structure of the file system from the Nandroid logical acquisition. Last but not least, a hex-editor can be very handy in checking for and extracting digital evidence which is not picked up by other tools.

IV. THE ANDROID ANTI-FORENSICS TECNIQUE

In this section, the proposed technique is described. It makes it possible to modify and delete in a secure and selective way the digital evidence on a device with the Android OS, without using any cryptographic additions or low-level modifications on the kernel.

Generally, the Android OS does not grant users access to the root of the file system. In order to gain access to the mtd5 partition on the internal flash memory as well as the /sd-ext on the microSD, and then modify/delete digital evidence on the device, it is essential to gain root privileges. There are a few guides on the Internet on how to gain root privileges to many Android devices. MoDaCo [11] and XDA-developers [12] are among the most active communities.

It is difficult to apply the anti-forensics techniques used for magnetic supports to NANDs flash memories. NANDs differ from hard drives in both the technology they use to store data as well as the algorithms they use to manage and access them. NANDs maintain a layer of indirection between the logical

block addresses that computer systems use to access data and the raw flash addresses which identify physical storage.

The layer of indirection enhances NAND performance and

reliability by both hiding the idiosyncratic interface of the flash memory and managing its limited lifetime. However, it can also produce copies of the data that are invisible to the user but that a sophisticated analysis can recover [7]. The differences between NANDs and hard drives therefore make it difficult to apply secure deletion and modification techniques used for magnetic supports to NANDs.

The authors did not find any papers in current literature

on the secure modification of flash memories. On the other hand, a few papers deal with the issue of secure deletion on flash memories. In [3] and [4], cryptographic primitives have been used to make the recovery of digital evidence difficult. However, the presence of cryptographic primitives is suspect in a forensics analysis. A technique which exploits the Android security features to delete digital evidence is presented in [5]. A third party application, at install time, creates a private folder used to contain possible subsequent digital evidence. This evidence is deleted during the uninstalling process of the application owning the private folder involved. According to [5] when a given common Android application is unin- stalled, the entire set of the related information, including data files and directories, is logically deleted from the file system.

The proposed anti-forensics technique has the following three characteristics:

• The use of only simple software tools which are quite commonly installed on devices running the Android OS. The use of cryptographic techniques and/or kernel modifications would have raised suspicion during a forensics analysis.

• The deletion is performed in a secure way by overwriting the data. It is also possible to use different sanitization techniques, for example those described in NIST 800-88 [13].

• The modification is performed without leaving any traces on the device. Moreover, the metadata and attribute list (e.g., timestamp, owner, etc.) can be preserved or changed in a suitable way, so as to not raise any suspicion during a forensics analysis.

The modification and deletion operations are performed in such a way that any previous copies of the data cannot be recovered or detected.

The key idea of the technique is to reduce the selective

and secure modification/deletion from a physical level on the

NAND to a logical level on the microSD.

The microSD is used as a temporary memorization support. It holds data contained in the internal NAND to allow for its subsequent sanitization. Afterwards, it is possible to selectively delete and modify data on the microSD using logical com- mands available on the device. Finally, data on the microSD is logically copied back to the internal flash memory. This copy acts only on the allocated files stored on the microSD, ignoring those present on the pages marked as deleted.



Fig. 1. Graphical rappresentation of the proposed technique.

Most of the Android devices collect in the branch/data both logs generated by the OS as well as those generated by third party applications. However, the file system structure is not the same for all the devices. For example, Android devices by Samsung collect these logs in the branch /dbdata. Even though the branches of the file system containing such logs are different, they are generally located on the partition mtd5. Therefore, herein after the description of the technique will refer to mtd5 and not to /data or /dbdata. After a suitable Setup phase, depending on the device and tools used, it is possible to apply the proposed technique. The technique consists of five steps, as shown in Fig. 1:

1) Copy data from mtd5 to mmcblk0p2: all the data, including the digital evidence, that has to be deleted or modified are copied to the partition mmcblk0p2 located on the microSD. The copy is on a logical level.

2) Sanitize mtd5: all the data in the partition mtd5 is erased using sanitization techniques.

3) Delete or modify digital evidence: the digital evidence which is located only in mmcblk0p2, is deleted or modified using logical operations.

4) Copy data from mmcblk0p2 to mtd5: all the data, including the digital evidence that has been modified, are copied onto the partition mtd5.

5) Sanitize mmcblk0p2: all the data in the partition

mmcblk0p2 is erased using sanitization techniques.

At the end of the five steps, the partition mtd5 contains only the data which were present on the pages marked as allocated on the partition mmcblk0p2 located on the microSD. In the data, there are the modified files targeted by the anti-forensic modifications. The digital evidence that was logically deleted in step 3) will not be present in the mtd5 partition since the logical copy operation in step 4) does not copy any of the pages marked as deleted. The partition mmcblk0p2 will be empty after the five steps. This is not suspicious for the following reasons. Starting from Android v2.2, an application can be installed on the microSD. It is also possible to move a previously installed application from the internal memory to the microSD provided that it has been formatted with the EXT3/EXT4 file system. Therefore, the presence of the partition mmcblk0p2 does not raise suspicion during a digital forensic analysis. If the device is running an older version of the Android OS, it is preferable to delete such a partition, since it can raise suspicion after a forensics analysis.

After the application of the proposed technique, there are no pages marked as deleted and containing duplicated blocks in the partitions mtd5 and mmcblk0p2. Generally, in flash memories there are multiple copies of some blocks. Such a difference could raise suspicions after a forensics analysis. Therefore, it is necessary to artfully produce deleted pages containing duplicated blocks. Since an anomaly detection analysis can distinguish artificial behaviour from real, natural

|  |  |  |  |
| --- | --- | --- | --- |
| Partition | Mount Point | Size |  |
| /dev/mtd/mtd0  /dev/mtd/mtd1  /dev/mtd/mtd2  /dev/mtd/mtd3  /dev/mtd/mtd4  /dev/mtd/mtd5 | -  -  -  /system  /cache  /data | -  -  -  145.0MB  95.0MB  196.3MB | Internal  Internal Internal Internal Internal Internal |
| /dev/block/mmcblk0p1  /dev/block/mmcblk0p2 | /sdcard  /sd-ext | 458.5MB  3.2GB | External  External |

TABLE I

behaviour, it is advisable to generate these pages so that they are similar to the normal Android OS and user behaviour. The content of the mtd5 partition should be changed to result undistinguishable from a real one by anomaly detection analysis. However, this analysis is very difficult to perform due to it having to consider both the users behaviour as well as the Android OS functionalities.

Another option that may be used to avoid the detection of the technique by an anomaly detection analysis is to apply some precautions at the beginning of the procedure. In fact, prior to applying the proposed technique, just after having per- formed the physical acquisition, it would have been collected unallocated pages. These pages should have been verified not being the result of the deletion/modification operations and brought back to their original position even in the resulting partition.

In order to verify the correctness of the proposed technique, a forensic investigation can be carried out by first obtaining the logical and physical acquisitions and then analyzing them, as illustrated in Section III. The analysis includes searching for the digital evidence, which has been deleted by the technique as well as any traces left by the technique itself. An unsuccessful search means a validation of the technique.

V. CASE STUDY

The focus of the case study is the application of the proposed technique to a specific device. This case study shows how the technique has been applied, and more generally gives useful advice on its application to different devices. A HTC Nexus One device equipped with a MIUI ROM was been used. The MIUI ROM is a modified Android v2.3.4 by the MIUI Team [14]. With this distribution, the user gains root privileges, complete access to the entire file system as well as the possibility to use the BusyBox tool. The BusyBox tool contains all the necessary software modules required for the operations involved in the case study, such as copy, delete, modify, overwrite (i.e. to run the commands cp, rm, touch, dd). The presence of the BusyBox tool installed on the device does not raise any suspicions since it is quite common to have it installed on an Android device. The structure of the internal flash memory, the microSD as well as the file system on the HTC Nexus One is summarized in Table I. The size column shows how big the partitions on the internal flash memory and microSD are.

A. Applying the Technique

The Setup phase includes the following two steps:

1) The reboot of the device in recovery mode. In fact, the copy, delete, modify and overwrite operations cannot be executed while the OS is running.

2) The verification of the presence of the partition mmcblk0p2 on the microSD. If this partition is not on the microSD it should be created. This partition has to be empty, with the same file system type of the mtd5 and not smaller. For example, if the mtd5 partition has been formatted with the EXT3/EXT4 file system type, then the mmcblk0p2 partition has to be similarly formatted. The use of the same file system type makes it possible to preserve the metadata and data attribute list during the copy process.

The description and analysis of the anti-forensics technique for the case study, consisting of the five steps shown in Fig. 1, is reported below.

Copy data from mtd5 to mmcblk0p2: The logical copy of the files from the partition mtd5 to the partition mmcblk0p2 has been implemented using the command: # cp -a /data /sd-ext

This command performs the copy by accessing the file system, ignoring the files present on the pages marked as deleted. The parameter -a allows for the recursive copy and preservation of the file attribute list (e.g., timestamp, metadata, etc.).

Sanitize mtd5: The secure deletion of the mtd5 partition is accomplished by overwriting all the pages, without considering whether they are marked allocated or deleted. The overwriting operation consists of setting each bit of the partition to “0”. Afterwards, the partition is formatted in the EXT3/EXT4 file system type. The commands used are the following:

# dd if=/dev/zero of=/dev/mtd/mtd5 bs=4096

# mke2fs -T ext4 -b 4096 -E stride=64, stripe-width=64

-O extent,ˆhuge\_file -m 0 -L userdata /dev/mtd/mtd5

It is also possible to use more sophisticated media sanitization techniques, as suggested in [13], by implementing each overwriting operation with an ad-hoc dd command.

Delete and modify digital evidence: Files on the partition mmcblk0p2 can be deleted and modified by using the logical functions implemented by the command of the BusyBox tool. In order to logically delete a file, the command rm can be used. To modify files, third party tools can be used. For example, using the MMS.apk, which is a preinstalled application in the Android OS, it is possible to delete a sent or received text message. To modify the timestamp of the database in order to synchronize it with the time of the last SMS present in database, the following command can be used: # touch -mt 201101052230.03 mmssms.db

The option -m makes it possible to set the last modification time of the database mmssms.db.

Copy data from mmcblk0p2 to mtd5: This is the same as the step Copy mtd5 to mmcblk0p2 in which the input and output are exchanged. The command used is:

# cp -a /sd-ext /data

Sanitize mmcblk0p2: This is the same as the step Sanitize mtd5 differing only in the input partition. The commands used are the following:

# dd if=/dev/zero of=/dev/block/mmcblk0p2 bs=4096

# mke2fs -T ext4 -b 4096 -E stride=64, stripe-width=64

-O extent,ˆhuge\_file -m 0 -L userdata /dev/block/mmcblk0p2

After the five steps, so as to avoid the lack of duplicated pages in the mtd5 partition, the authors carried out the following operations:

• send and receive emails;

• surfed on the Internet (this generates many cache files);

• send and receive text messages;

• originate some calls.

B. Technique Validation

The technique described and implemented in the case study was validated by running multiple experiments, which in- cluded many modifications and/or deletions. This was followed by performing both physical and logical acquisitions as well as a forensic analysis on them. During the verification phase, all the open-source tools mentioned in Section III, along with the demo version of Oxygen Forensics Suite 2011, were used. In order to check the effectiveness of the file deletion opera- tion, several files containing predetermined short patterns were created on the flash memories used. After the application of the technique, the presence of these patterns was searched for. The forensic copies of the internal and external flash memories were transferred to an external PC and inspected with a hex-editor so as to perform the search of the patterns. In addition, to further test the correctness of the deletion of the digital evidence, digital forensics open-source tools (Autopsy and PhotoREC) were used.

In order to validate the anti-forensics technique used to modify files on the device, it was applied to different files several times. All the modifications were carried out on the device without requiring any external hardware. Some modifi- cations were made using the programs that were shipped with the equipment, while others by using third party applications. Some of these tools were mentioned in Section V. Most of the modifications focused on system applications such as logs, text messages, calls, contacts, emails, etc.. After having applied the modifications, all the files and applications involved were checked to verify whether the changes had been applied or not and the timestamps consistency. Moreover, keeping in mind that flash memories may duplicate some parts of the data, a check was carried out on whether

any duplications of the modified information, referring to an unmodified version of it, was found on the device.

All the forensics analysis performed have confirmed the validity of the technique described.

VI. CONCLUSIONS AND FUTURE WORKS

A new anti-forensics technique for the Android OS was proposed and analyzed. By using the latest digital forensics tools, and considering the specific characteristics of the flash memories used by Android devices, the technique was vali- dated by showing its effectiveness in a case study. While the device adopted in the case study was a specific mobile phone running the Android OS, the authors plan to investigate their technique further on different kinds of Android device (tablet, TV, netbook). Moreover, considering that the type of memories used by the majority of the Android devices are the same as those used by most mobile devices, a future study could be carried out on devices using similar types of memories (e.g., Apple iPhone). It would be interesting to characterize the distribution of the duplicated pages in order to simulate it as well as pass any eventual anomaly detection inspections.

REFERENCES

[1] Gartner Inc. and/or its Affiliates, “Gartner Analysts,” [http://www.gartner. com/technology/analysts.jsp, 2011](http://www.gartner.com/technology/analysts.jsp) (accessed May 09, 2011).

[2] R. Harris, “Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem,” in The 6th Annual Digital Forensic Research Workshop (DFRWS 2006), Aug. 2006. [Online]. Available: <http://dfrws.org/2006/proceedings/6-Harris.pdf>

[3] D. W. Byunghee Lee, Kyungho Son and S. Kim, “Secure data deletion for usb flash memory,” Journal of Information Science and Engineering, pp. 1710–1714, 2011.

[4] J. Lee, J. Heo, Y. Cho, J. Hong, and S. Y. Shin, “Secure deletion for nand flash file system,” in ACM Symposium on Applied Computing, 2008, pp.

1710–1714.

[5] A. Distefano, G. Me, and F. Pace, “Android anti-forensics through a local paradigm,” Digital Investigation, vol. 7, pp. S83–S94, Aug. 2010. [Online]. Available: <http://dx.doi.org/10.1016/j.diin.2010.05.011>

[6] Charles Manning, “YAFFS: the NAND-specific flash file system - Introductory Article,” <http://www.yaffs.net/yaffs-nand-specific-flash-file-> system-introductory-article, 2011 (accessed May 27, 2011).

[7] M. Y. C. Wei, L. M. Grupp, F. E. Spada, and S. Swanson, “Reliably erasing data from flash-based solid state drives,” in FAST, G. R. Ganger and J. Wilkes, Eds. USENIX, 2011, pp. 105–117.

[8] J. Lessard and G. C. Kessler, “Android forensics: Simplifying cell phone examinations,” Small scale digital device forensics journal, Aug. 2010.

[9] viaForensics, “viaForensics’ AFLogical Tool is Best for Android Forensic Investigations,” <http://viaforensics.com/viaforensics-> articles/viaforensics-aflogical-tool-android-forensic-investigations.html,

2011 (accessed June 22, 2011).

[10] Kai.Wei.cn, “Unyaffs project,” [http://code.google.com/p/unyaffs/, 2011](http://code.google.com/p/unyaffs/) (accessed May 27, 2011).

[11] P. O’Brien, “Android @ modaco.com,” [http://android.modaco.com/,](http://android.modaco.com/)

2011 (accessed May 27, 2011).

[12] XDAdevelopers, “Android forum & windows phone discussion,” [http:](http://forum.xda-developers.com/index.php)

[//forum.xda-developers.com/index.php, 2011](http://forum.xda-developers.com/index.php) (accessed May 27, 2011). [13] NIST, “NIST 800-88, Guidelines for Media Sanitization,” [http:](http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_rev1.pdf)

[//csrc.nist.gov/publications/nistpubs/800-88/NISTSP800- 88 rev1.pdf,](http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_rev1.pdf)

2011 (accessed May 27, 2011).

[14] MIUI Team, “MIUI Android ROM,” [http://www.miui.com, 2011](http://www.miui.com) (ac- cessed June 20, 2011).