Mobile Phone Anomalous Behaviour Detection for Real-time Information Theft Tracking

Vrizlynn L. L. Thing[1], Perumal P. Subramaniam[2], Flora S. Tsai[2], and Tong-Wei Chua[1]

[1]Cryptography & Security Department
Institute for Infocomm Research, Singapore
{vriz,twchua}@i2r.a-star.edu.sg
[2]School of Electrical and Electronic Engineering
Nanyang Technological University
peru0002@ntu.edu.sg, fst1@columbia.edu

***Abstract*—Due to the prevalence of mobile phones usage and their increasing features and functionalities, the amount of personal and confidential data residing in the phones is becoming substantial. In the event of information theft by applications residing on the phones, the loss of such important data can be damaging to the user's reputation or result in a financial loss. We show in this paper how these applications can appear to be non-malicious but are stealthily retrieving and exporting confidential information without leaving any trace, thus bypassing detections by current state-of-the-art anti-virus solutions. We propose a tool to detect and track the behaviour of these applications in real-time so as to collect evidence. Using this tool, we can successfully monitor the applications non-intrusively, detect the "misbehaving" applications, alert the users, and log the evidence of malicious activities with timestamp information to facilitate forensic investigations and institute accountability.**

Keywords: Mobile device forensics, Android, information theft, anomaly detection, spyware.

I. Introduction

As mobile phones are becoming increasingly prevalent and are constantly evolving into "smarter" devices (i.e. smartphones with higher processing power and enhanced features), it is a common scenario that users are installing applications that appeal to them while at the same time generating and storing more personal information on their phones. In the presence of information theft activities introduced by the installed applications, there is a risk of users losing their personal and sensitive information, or confidential corporate data (e.g. emails).

Due to the potential risks, the capabilities to perform detection of such malicious information theft activities on the phones and the collection of relevant evidence to facilitate forensic investigation become essential. However, current mobile phone forensics are still restricted to the research and analysis of static data on the Subscriber Identity Module (SIM), memory cards and the internal flash memory [1], [2], [4], [5], [8], [10]–[13], [16], [19].

To achieve anti-theft protection and anti-virus scanning on mobile phones and devices, there exist several mobile devices security solutions [6], [7], [9], [15], [18] in the market. The features of these solutions include blocking the phones, deleting the data and finding the phone (via GPS and map display) through remote access by the user, in the event that the user loses his/her phone. They also support the detection and removal of malicious applications from the phones. However, the detection of virus is through a signature based mechanism and do not support anomaly detection or real-time monitoring for information theft protection against malicious applications. More information on the related work is discussed in Section II.

In this paper, we propose a tool to monitor, detect, and track cyber criminal activities relating to information thefts on the mobile phones or devices. The tool intercepts sensitive information access non-intrusively (i.e. without interfering with normal application operations), triggers an alert and performs evidence collection (i.e. logging and timestamping) for mobile devices. In our work, we focus on mobile phones running the Android operating system. The reason is that while there exists the availability of a high number of applications in the Android Market online portal, the market place is not well regulated, unlike the iPhone App Store. Applications written by third party developers are not required to be approved before being available for download by mobile phone users. This scenario presents a potential risk in malicious activities being introduced to the phones. In addition, Android is a new mobile platform but its fast rising popularity among users and the phone manufacturers [14] calls for a need for us to address the security and forensic issues with regards to the information theft problem.

The rest of the paper is organised as follow. In Section II, we present an overview of the related work in the area of mobile phone anti-spyware. We describe the design of our detection and tracking tool in Section III. The experiments and results are presented and discussed in Section IV. Future work is described in Section V and conclusions follow in Section VI.

II. Related Work

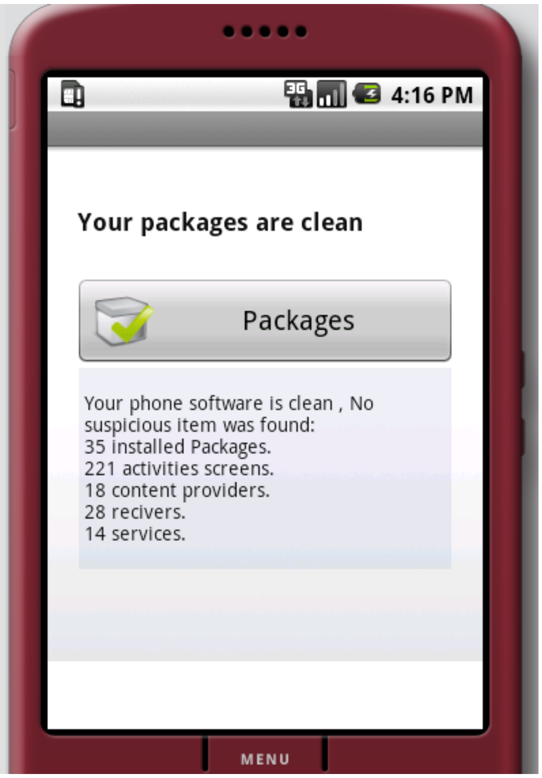
In the personal computer terminology, malicious applications or software include viruses, worms and other exploits. Such malicious applications or software can also exist in mobile phones, and can be exploited by criminals to steal

ISBN: 978-1-61208-122-9

sensitive information from the phones' users discreetly. In this section, we discuss the current state-of-the-art mobile security solutions.

There are a number of anti-virus software in the market for mobile devices, such as the Kaspersky Mobile Security [15] and Norton Smartphone Security [7]. They support the Symbian and Windows mobile platforms. Mobile security solutions for Android include the Droid Security [9], SMobile Mobile Security [18] and F-Secure Mobile Security [6]. The Droid Security tool enables the user to remotely block and delete the data on the phone in the event that the phone is lost. Other features include the GPS locator, virus scanning and the identification of dangerous websites. Similar features are supported by the SMobile Mobile Security and the F-Secure Mobile Security solutions.

We tested the freely available trial version of the Droid Security solution to detect the presence of a malicious application on the Android phone. We built the malicious application based on the code snippet of a simple tips calculator [17] by modifying it to access the phone contacts information and exporting it to an external party in the background. Therefore, in the foreground, the application appears to be an innocent looking application performing basic calculator operations but is in fact carrying out information theft activities stealthily.



Your packages are clean

Your phone software is clean , No suspicious item was found:
35 installed Packages.
221 activities screens.
18 content providers.
28 recivers.
14 services.

Fig. 1: Scan results with Droid Security

When using the Droid Security software to scan for malicious applications, it could not detect the tips calculator as potentially harmful to the user. Figure 1 shows the results of scanning the phone with the Droid Security. All the installed applications were found to be clean.

Through this small experiment, we observed that although existing mobile security solutions can provide protection from information theft due to the physical loss of the mobile phones or devices and support virus scanning to detect and remove known malicious software, there is a need for a solution to monitor, detect, alert and collect evidence of information theft (which is conducted stealthily) on the mobile phones in real-time so as to facilitate the forensic investigations of such violations by the law enforcement agencies. Therefore, in this paper, we design a real-time information theft detection and application behaviour tracking tool, and present it in the next section.

III. Information Theft Detection and Tracking

Most of the applications for Android are available in the Android Market which is an online store for Android applications. Users browse and download applications published by third-party developers as an open service and this process is not well regulated [3]. Thus, it is difficult to determine whether a newly available application is malicious or not. To perform a real-time monitoring and detection of the anomalous and suspicious behaviour of installed applications, we propose designing a tool that intercepts selected applications' access to sensitive information and alerts the user of their intentions.

A. API Hooking

The Application Programming Interface (API) is an interface which is used by an application to request services from libraries and operating systems. Hooking is a technique where the normal program flow is diverted. API Hooking is a procedure where the API calls by programs to interact with the kernel are intercepted. Figure 2 shows an overview of API hooking.

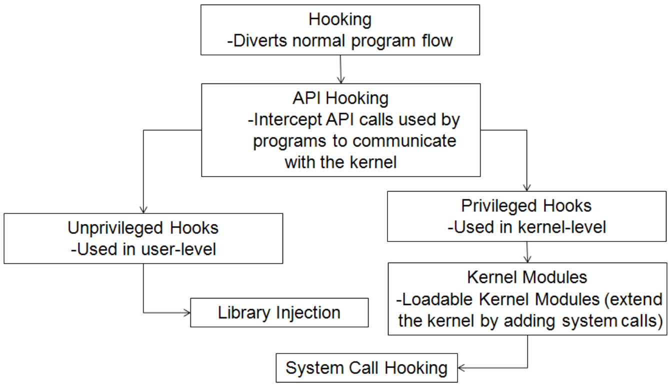

Hooking
-Diverts normal program flow
API Hooking
-Intercept API calls used by programs to communicate with the kernel
Unprivileged Hooks
-Used in user-level
Privileged Hooks
-Used in kernel-level
Library Injection
Kernel Modules
-Loadable Kernel Modules (extend the kernel by adding system calls)
System Call Hooking

Fig. 2: API Hooking

There are two types of hooks, unprivileged and privileged hooks. Unprivileged hooks can only be executed within the user program address space through library injections while privileged hooks can be executed in kernel level to replace or add on to system calls.

B. Tool Design

In Android, the applications access the kernel through "legal entry points", which are known as the system calls. The

ISBN: 978-1-61208-122-9

system calls enable the mobile phone applications to access the kernel while maintaining the system's stability. They provide an interface for a user-space process to request for operating services such as specific accesses to different hardwares or fundamental services (e.g. generic system read and write as shown in Figure 3).

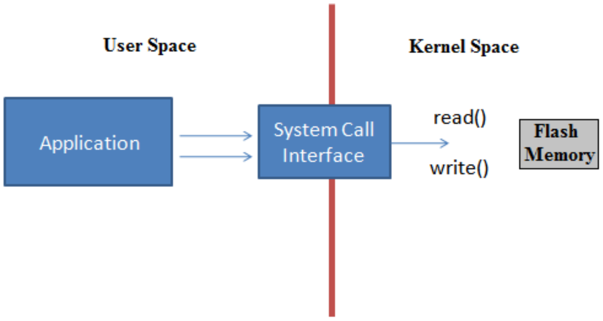


Fig. 3: System Call

System call interception is an example of a privileged hook (3). Even though a malicious application can function discreetly on the phone, its operating system service access can still be intercepted so as to facilitate the monitoring of its actual underlying activities. Therefore, to detect any suspicious theft activities, we build a loadable kernel module to observe the user selected applications and detect any request for services from the operating system to access sensitive and confidential information on the phone. This monitoring module "captures" the system calls to support our tool's analysis of the activities undertaken by the applications.

To perform system calls interception on the phone, we accessed the system call table to locate the system call references. The system call table address at 0xc0539900 was revealed through the "print &sys_call_table" call. After which, we added the functions to intercept specific system calls (i.e. system access, network socket calls, folder deletion, change of ownership rights). The program flow can also be diverted automatically if the calls are found to be malicious (Figure 4). Our tool binds to an application upon selection and non-intrusively monitors its activities and alerts upon access to sensitive information on the phone. In a compromised application, the tool can detect the anomalies even if the application "misbehaves" in the background.

IV. Experiments and Discussions

For our experiments, we configured our tool as follow.

1) Observe and log all important system calls (i.e. system access, network socket calls, folder deletion, change of ownership rights)
2) Specify the alert conditions (e.g. read/write access to contacts information) to detect suspicious activities and possible information theft, and to inform users in real-time (i.e. display floating messages over the applications)
3) Log suspicious activities with timestamp information in the background for evidence collection and forensic investigations

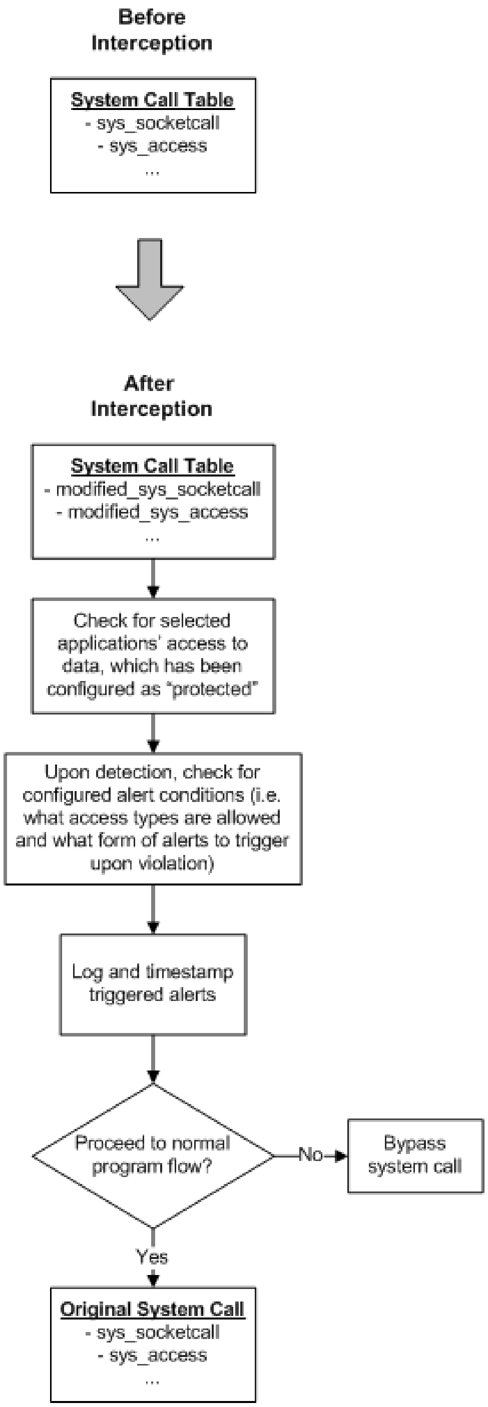
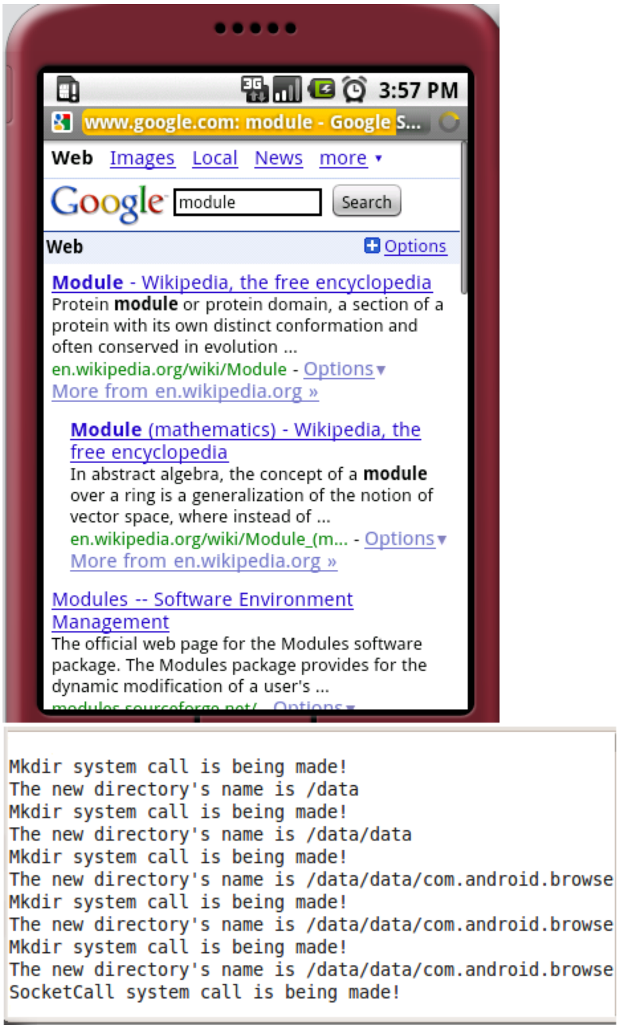

Before Interception
System Call Table
- sys_socketcall
- sys_access
...
After Interception
System Call Table
- modified_sys_socketcall
- modified_sys_access
...
Check for selected applications' access to data, which has been configured as "protected"
Upon detection, check for configured alert conditions (i.e. what access types are allowed and what form of alerts to trigger upon violation)
Log and timestamp triggered alerts
Proceed to normal program flow?
No
Bypass system call
Yes
Original System Call
- sys_socketcall
- sys_access
...

Fig. 4: Tool Design

We conducted two experiments. The first was an experiment to track the behaviour of an Internet browser application on the Android phone. The second experiment was conducted on the "misbehaving" tips calculator mentioned in Section II.

In the first experiment, we selected the default Internet browser on the phone to be started. Our tool will advise the user to load the interception module so as to bind to the application. The user then started using the browser as usual while the behaviour tracking was conducted in the background (refer to Figure 5). As the application was not accessing any


3:57 PM
www.google.com: module - Google S...
Web Images Local News more ▾
Google
module
Search
Web
Options

Module - Wikipedia, the free encyclopedia
Protein **module** or protein domain, a section of a protein with its own distinct conformation and often conserved in evolution ...
en.wikipedia.org/wiki/Module - Options ▾
More from en.wikipedia.org »

Module (mathematics) - Wikipedia, the free encyclopedia
In abstract algebra, the concept of a **module** over a ring is a generalization of the notion of vector space, where instead of ...
en.wikipedia.org/wiki/Module_(m... - Options ▾
More from en.wikipedia.org »

Modules -- Software Environment Management
The official web page for the Modules software package. The Modules package provides for the dynamic modification of a user's ...

```
Mkdir system call is being made!
The new directory's name is /data
Mkdir system call is being made!
The new directory's name is /data/data
Mkdir system call is being made!
The new directory's name is /data/data/com.android.browse
Mkdir system call is being made!
The new directory's name is /data/data/com.android.browse
Mkdir system call is being made!
The new directory's name is /data/data/com.android.browse
SocketCall system call is being made!
```

Fig. 5: Behaviour Tracking on Phone

private user information, no alert was triggered in this case.

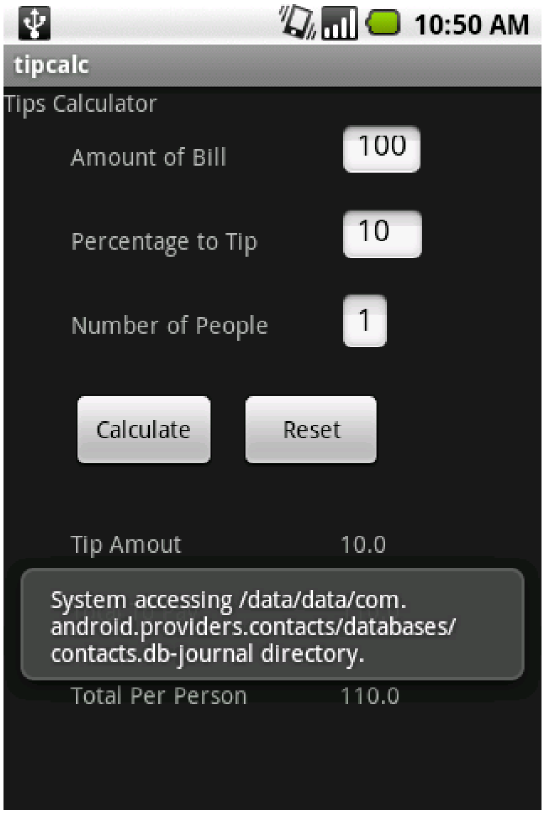
In the second experiment, we first run the tips calculator application without activating our tool. After the application was started, it accessed the phone contacts information stealthily in the background while the user was using the application. The contacts information on the Android phone was successfully sent to the external party phone through SMSes. We performed a manual investigation of the "sent messages" records on the phone and found that no trace of the export could be found.

We closed the application and started it again; this time with our tool activated. In this case, the following alert was triggered.

```
Alert: 12:20 PM April 12, 2010 System
accessing data/data/com.android.
providers.contacts/databases/contacts.
db-journal directory.
```

The log file was updated and the floating message was also displayed over the application to inform the user (refer to Figure 6).

As shown in the experiments, the detection and tracking tool can reliably capture suspicious activities due to the access to secure information, in real-time. These activities are logged and timestamped, and stored in the phone's non-volatile memory to enable further forensic analysis. The user is also alerted of the ongoing underlying suspicious activities carried out by the application through a real-time display of warning messages. This enables the user to be aware of potentially harmful applications even without any prior knowledge that the application is malicious.



Amount of Bill 100

Percentage to Tip 10

Number of People 1

Tip Amout 10.0

System accessing /data/data/com.android.providers.contacts/databases/contacts.db-journal directory.

Total Per Person 110.0

Fig. 6: Alert Message Display

V. Future Work

Our planned future work includes implementing further enhancements to the tool to include the monitoring of relevant Android application interfaces such as the radio interface layer. This would enable the support of a more detailed logging of information such as the phone number to which the stolen data is exported to or which server is accessed by the malicious application. Another enhancement includes the automatic exporting of logged evidence to a server to support further forensic investigation. The displayed alert messages could also be made more understandable (i.e. less technical) to the common users.

VI. Conclusions

As mobile phones become more prevalent with increasing personal sensitive information and possibly confidential corporate data (e.g. accessed through emails) generated or downloaded on to the phones, there exists a potential risk of losing important information due to malicious information theft exploits. With the growing size of the available applications in the Android market, there is also an increased chance of a malicious application being installed successfully by the users unknowingly. The high possibility of potential mobile information theft crimes call for the need to address the security and forensic issues to institute accountability.

Although there exist anti-virus solutions in the market for mobile phones and devices, malicious information theft applications can bypass their detection by remaining stealth at the application level. Since access to the confidential data is an essential step to stealing it, we non-intrusively intercept the system services access, track the behaviour of the applications and alert the user (and record such violations) when the applications "misbehave" by accessing the data they are not supposed to. We implemented our tool for the Android mobile operating system platform due to its rising popularity and its open service online application market. We showed through experiments that even though no SMS record of the stolen data export could be found during a manual investigation, our tool can reliably detect and track the malicious access in real-time. Our tool is also flexible and highly configurable to indicate which system calls to observe and which data access will trigger alerts.

REFERENCES

[1] Rizwan Ahmed and Rajiv v. Dharaskar. Mobile forensics: an overview, tools, future trends and challenges from law enforcement perspective. *6th International Conference on E-Governance, ICEG, Emerging Technologies in E-Government, M-Government*, pages 312–323, December 2008.

[2] Marwan Al-Zarouni. Mobile handset forensic evidence: a challenge for law enforcement. In *Proceedings of the 4th Australian Digital Forensics Conference*, December 2006.

[3] Android. Android developers. http://developer.android.com/index.html.

[4] Rick Ayers, Wayne Jansen, Ludovic Moenner, and Aurelien Delaitre. Cell phone forensic tools: An overview and analysis update. *National Institute of Standards and Technology, Technical Report 7387*, March 2007.

[5] Fabio Casadei, Antonio Savoldi, and Paolo Gubian. Forensics and SIM cards: an overview. *International Journal of Digital Evidence*, 5(1):1–21, Fall 2006.

[6] F-Secure Corporation. F-secure Mobile Security. http://campaigns.f-secure.com/mobile-security/index.html.

[7] Symantec Corporation. Norton Smartphone Security. http://www.symantec.com/norton/smartphone-security.

[8] Alessandro Distefano and Gianluigi Me. An overall assessment of mobile internal acquisition tool. *Proceedings of the 8th Digital Forensics Research Conference (DFRWS), Digital Investigation*, 5(1):S121–S127, September 2008.

[9] droidSecurity. Droid Security - Anti-virus. http://www.droidsecurity.com/.

[10] Andrew Hoog. Android forensics. *presented at Mobile Forensics World 2009*, May 2009.

[11] Andrew Hoog and Kyle Gaffaney. iPhone forensics. *viaForensics Whitepaper*, June 2009.

[12] Wayne Jansen and Rick Ayers. Forensic software tools for cell phone subscriber identity modules. *Conference on Digital Forensics, Association of Digital Forensics, Security, and Law (ADFSL)*, April 2006.

[13] Wayne Jansen, Aurelien Delaitre, and Ludovic Moenner. Overcoming impediments to cell phone forensics. In *Proceedings of the 41st Hawaii International Conference on System Sciences*, 2008.

[14] Greg Kumparak. Google: Android now shipping on 60,000 handsets per day. http://www.mobilecrunch.com, February 2010.

[15] Kaspersky Lab. Kaspersky Mobile Security. http://www.kaspersky.com/kaspersky_mobile_security.

[16] Pontjho M. Mokhonoana and Martin S. Olivier. Acquisition of a Symbian smart phone's content with an on-phone forensic tool. *Department of Computer Science, University of Pretoria*, 2007.

[17] Data Springs. A brief guide for creating your first android application (tip calculator). http://www.datasprings.com/Resources/ArticlesInformation/AndroidSDKExampleApplicationSampleCode.aspx.

[18] SMobile Systems. SMobile Mobile Security. http://www.smobilesystems.com/.

[19] Svein Willassen. Forensic analysis of mobile phone internal memory. *Advances in Digital Forensics, IFIP International Federation for Information Processing, Springer*, 194:191–204, March 2006.

ISBN: 978-1-61208-122-9