The Forensic Analysis of a False Digital Alibi

Aniello Castiglione*, Giuseppe Cattaneo†,
Giancarlo De Maio‡, Alfredo De Santis§
Dipartimento di Informatica, Università degli Studi di Salerno
I-84084 Fisciano (SA), Italy
castiglione@acm.org*, cattaneo@dia.unisa.it†,
demaio@dia.unisa.it‡, ads@dia.unisa.it§

Gerardo Costabile¶, Mattia Epifani‖
International Information Systems Forensics Association
IISFA Italian Chapter
Via Tibullo 11, I-00193 Roma, Italy
gerardo@costabile.net¶, mattia.epifani@digital-forensics.it‖

***Abstract*—In recent years the relevance of digital evidence in Courts disputes is growing up and many cases have been solved thanks to digital traces that addressed investigations on the right way. Actually in some cases digital evidence represented the only proof of the innocence of the accused. In such a case this information constitutes a digital alibi. It usually consists of a set of local and Internet activities performed through a digital device. It has been recently shown how it is possible to setup a common PC in order to produce digital evidence in an automatic and systematic manner. Such traces are indistinguishable upon a forensic post-mortem analysis from those left by human activity, thus being exploitable to forge a digital alibi.**

In this paper we verify the undetectability of a false digital alibi by setting up a challenge. An alibi maker team set up a script which simulated some human activities as well as a procedure to remove all the traces of the automation including itself. The verification team received the script and executed it on its own PCs. The verification team could perform not only a usual post-mortem analysis but also a deeper forensic analysis. Indeed, they knew all the details of the script and the original state of the PC before running it. The verification confirmed that a well-constructed false digital alibi is indistinguishable from an alibi based on human activities.

***Index Terms*—Digital Evidence; Digital Investigation; Digital Forensics; Anti-Forensics; Counter-Forensics; Automated Alibi; False Alibi; Digital Alibi; False Digital Alibi.**

I. Introduction

Lives of people are increasingly being dependent on technology. They rely on digital devices for entertainment, to communicate each other, to store sensitive information, to manage their assets, to buy products and so on. More than two billion people use Internet, with a penetration higher than 30% (statistics for December 31, 2011) [1]. Furthermore, traditional security paradigms based on the strong separation between a secure "inside" area and a hostile "outside" one, are no longer effective in presence of modern borderless communication infrastructures [2].

In this context, more and more crimes are performed on the Internet or have something to do with digital equipment. For these reasons the amount of digital evidence being brought in Courts is rapidly increasing. Consequently, Courts are becoming concerned about the admissibility and probative value of digital evidence. Although digital devices have not been directly used by an accused, they can be subject to forensic investigations in order to collect useful traces about the behavior of such individual, either to be cleared of an accuse or to be charged of an offense. Elements required to determine the liability for having committed a crime often consist of files stored in a PC memory, photos on a digital camera, data on a mobile phone or any other digital devices. Such elements are referred to digital evidence. "Digital evidence or electronic evidence is any probative information stored or transmitted digitally and a party to a judicial dispute in Court can use the same during the trial" [3]. E-mails, digital photos, electronic documents, instant message histories, spreadsheets, Internet browser histories, databases, temporary files and backups can contain useful information in the context of a legal proceeding to reconstruct the profile and the user behavior in a certain time frame of a defendant.

Those traces are *ubiquitous*, *immaterial* and *implicit*, i.e., in many cases the individual who "generated" them is unaware of their existence. In fact, digital traces can be retrieved on mobile devices (phones, PDAs, laptops, GPSs, etc.) but especially on servers that provide services via Internet, which often register the IP addresses and other information concerning the connected clients. These servers can be located in countries different from the one in which the crime has been committed, and different national laws can be an obstacle for the acquisition of the digital evidence during an investigation. Such digital traces are also immaterial because it is well known that all digital data is represented as sequences of binary digits. Obviously, digital data can be easily manipulated.

This paper aims to state how a deep forensic analysis is able to discern whether the evidence collected from a digital device might have been artificially created. In particular, this problem has been considered in order to ascertain whether the methodology introduced in [4] effectively weaken the acceptability of a digital alibi in Courts. Early experiments showed that digital evidence constructed using such a methodology is indistinguishable from the one produced by a human activity. For that reason, this paper presents new experiments where the forensic analysts have more knowledge about the methodological and technical procedures employed to create the false alibi. In details, the authors setup a challenge between two groups. The first one prepared the procedure while the other one performed the forensic analysis of the system where the procedure was

Corresponding author: Giancarlo De Maio, demaio@dia.unisa.it, Dipartimento di Informatica, Università degli Studi di Salerno, Via Ponte don Melillo, I-84084 Fisciano (SA), ITALY. Phone: +39089969594, Fax: +39089969821

run. The former designed a storyboard with all the activities to be executed in unattended way creating an automation script. Successively, this was sent to the other group in order to create the digital evidence. After the execution, the PC where the automation was run has been scrutinized by the analysts to detect, as much as possible, anomalous traces proving that all the evidence have been produced by an automatic procedure without requiring the presence of an individual in the alibi time frame.

The paper starts with a brief presentation of the concept of false digital alibi in Section II. The core idea of the paper, including the challenge organization and details about the Digital Forensics methodology used for the analysis, is presented in Section III. A case study showing an implementation of the aforementioned methodology and its results is presented in Section IV. Conclusions are drawn in Section V.

II. False Digital Alibi

Computers can be involved in the commission of crimes and can contain evidence of crimes, but these evidence can also represent an *alibi* for the defense of an accused person. In the Latin language the word "*alibi*" is an adverb meaning "*in or at another place*". According to the Merriam-Webster online dictionary [5], alibi is "the plea of having been at the time of the commission of an act elsewhere than at the place of commission".

a) False Digital Alibi: In a recent paper [4] a methodology to produce a predetermined set of digital evidence by means of an automation has been presented. Such evidence resulted to be indistinguishable, upon a post-mortem forensic analysis, from those produced by the real user activity. It demonstrates that this methodology could be exploited in the context of a legal proceeding to forge a digital alibi. Any typical actions performed by an individual on a computer can be simulated by means of automated tools, including mouse clicks, pressure of keys, writing of texts, use of applications, and so on. When a false digital alibi is forged might be worth to automate, among the other actions, the access to Internet public services because, in these cases, the alibi could also be confirmed by the digital evidence collected on the remote servers and validated by the remote service providers as trusted third parties. However, people have their habits and follow a predictable behavioral patterns. For example, it may be usual for an individual to read news during the morning, to access his mailbox at certain time intervals, to work on some documents in the afternoon, to shutdown the computer in the evening. In practice, the behaviour of the suspect inferred from his digital alibi must be very similar to his typical behaviour. Moreover, no suspicious traces have to be discovered by an hypothetical Anomaly Detection analysis [6].

A similar approach, presented in [4], can be also adopted in order to produce indistinguishable digital evidence by means of a mobile device equipped with Android [7] [8].

b) Unwanted Evidence: An automation can leave clues revealing its presence on the system. Such a trace is referred to as unwanted evidence, and should be always *avoided* or *removed* in order to divert the forensics analysis. There are basically two approaches that can be adopted to accomplish this task:

- *Avoid evidence a-priori.* Several precautions can be taken in order to avoid as much unwanted evidence as possible. Whenever it is not possible to securely remove an unwanted trace (for example, when its location is write-protected), an *a-priori obfuscation* strategy could be adopted in order to avoid any logical connections between this information and the automation procedure, in a way that such information could have been produced by a "normal" system operation.
- *Remove evidence a-posteriori.* In the last decade a large number of wiping techniques [9] [10] [11] have been proposed. Although they can be adopted to remove unwanted data left by the automation on a memory, in this context the actual problem is "how to erase the eraser". In [12] several methods that can be exploited to implement an automatic, selective and secure deletion/self-deletion are shown.

c) Development and Testing: The construction of an automation consists of two iterative phases: (1.) the development of the automation and (2.) the testing of the procedure on the target system. These activities could leave lot of clues on the target system which should be avoided. The ideal solution would be that of carrying out such tasks on a completely separate system, like a secondary computer or a Virtual Machine (VM). Otherwise, any unwanted evidence has to be detected and removed/obfuscated as described before. Typical unwanted traces produced during the construction stage are system records and logs, which usually contain recently opened files and launched applications. For example, in Microsoft OSes information like that is stored in the Windows Registry, which can only be modified by the *administrator* and the modifications take effect only after a system reboot.

In [4] a number of workarounds to avoid most of the suspicious traces about the development phase are proposed. In order to minimize the risk of unwanted clues, it could be useful to develop the automation on a completely separate environment, such as a VM, a live OS, or even a different computer. Since an automation is typically strictly cut for a specific hardware/software environment, the development system should be as similar as possible to the target system.

III. Forensic Analysis as a Challenge

Scientific Knowledge must be objective, reliable, provable and understandable. In other words, a thesis should always be supported by experimental results and such experiments must be repeatable by everyone and, in the same conditions, should produce always the same results. Keeping that in mind, the authors have intended to apply this approach to prove their thesis.

To validate the methodology presented in [4] it is crucial that the verification procedure can be carried out by a competent third part. In the occasion of the symposium UAE OLAF 2011 [13], held in Milano in January 2011, a challenge has been conducted among the authors of [4] and an IISFA [14] analyst team. The experiment consisted of the post-mortem

analysis of the hard disk extracted from a PC where a false alibi procedure was run. The goal of the analysis focused on the identification of any digital evidence left in a fixed time frame. Moreover, the analysts had to verify whether such evidence has been produced by a human activity or a software automation. The analysts where able to completely reconstruct the whole activities timeline. No digital evidence of automation execution, except a statistical anomaly, were found. In facts, only an anomalous browsing activity of a photo album on Picasa was identified. In particular, the analysts detected that 10 photos were requested with a constant interval of about 5780 milliseconds [15]. This was due to the presence of an iterative block of repeated actions where the time delay did not change. Apart from such suspicious evidence, the challenge confirmed that a false digital alibi is indistinguishable in a post-mortem forensic analysis.

In order to conduct a deeper analysis on the eventual traces left by the automation procedure, the following scenario has been considered. The analysts had complete information of the entire alibi-construction process (including the source code of the automation) and were able to run the automation at their will. Hence the analysts could choose the right tools and procedures, without being limited by the post-mortem approach. As an example, they had knowledge of the system status before and after the execution of the automation, so they could automatically compute all the relative differences. For a deeper analysis, a new challenge has been arranged as described in the following of the paper.

It is important to highlight that an automation resistant against the analysis presented in this work is certainly resistant against a forensic analysis conducted in a real context, since the evidence available to the analysts is certainly a "superset" of the evidence that would be available in a real case. In fact:

- The analysts have full knowledge of the methodology adopted to construct the alibi and any implementation details.
- The analysts could themselves setup the testing environment (according to a few required settings) on their computers.
- The analysts could execute the automation as many times as they wish.
- The analysts could record different system snapshots useful to detect any artifacts left by the automation. For instance, they could search for traces of the automation files (e.g., alphanumeric substrings) on the system memories, as well as identify any files modified upon the execution of the automation.

A. Organization of the Challenge

A challenge between two teams, the Alibi Makers (AM) and the Digital Forensics Analysts (DFA), has been arranged. The following methodology has been adopted to conduct the analysis: the AM constructed an automated procedure P according to [4] in order to forge a digital alibi by means of a common computer. This step has been carried out on a development system. The automation, including documentation and source codes, has been transferred to the DFA. The DFA setup a Testing System (TS) where the procedure P could be executed. The DFA finally analyzed the TS in order to collect any eventual traces supporting the thesis that the alibi has been produced by an automation.

Initially, the two teams agreed on the operational environment. This was a non-trivial task because they had to prearrange the TS hardware and software configuration in order to simulate a real scenario. The TS was equipped with:

- common hardware resources (i.e., storage device such as magnetic hard disk, standard input devices such as mouse and keyboard, etc.),
- a common software environment (i.e., the same OS, widely used applications, common libraries, etc.).

Clearly, these choices influence the software tools and the wiping methods to be used for developing the alibi.

A crucial aspect is how the automation files are imported on the target machine. In fact, a large amount of unwanted traces could be produced at this stage. For example, the OS could log information about the device used to transfer the automation to the computer, such as a CD-ROM or an USB drive, as well as information about the network service used to download the respective files.

The AM and DFA also dealt with the problem of the removal of the automation from the TS. It is important to note that the typical delete operation performed by means of the OS system calls is not enough to remove any traces of a file from the system. In order to avoid data remanence, a secure deletion procedure should be adopted. In this work, *secure deletion* is referred to as a procedure able to both wipe the target files (i.e., overwrite the physical disk blocks storing their data) and erase any traces of its presence from the system (i.e., data and metadata referring to the file containing the procedure itself). In other words, the wiper should self-delete its image on disk.

Among the various solutions to this problem proposed in [12], in this work a technique based on the use of a Secure Digital (SD) memory card has been chosen. More precisely, the automation has been loaded onto a SD subsequently connected to the TS by means of an USB adapter. Even though USB memories can be used as a secure authentication means [16], for the purpose of this paper the USB technology is used as an hardware proxy with the primary goal of hiding the object inserted into it (i.e., the SD). In fact, only the *device-ID* (i.e., the pair VID and PID) of the adapter is logged by the hosting machine, with no reference to the contained object. Afterward, the automation has been executed without copying it onto the hard-disk of the TS, thus avoiding the risk of data remanence on its main filesystem. The secure deletion procedure designed for the challenge firstly wipes any automation files, then fills up the residual SD space in order to overwrite unused memory blocks eventually containing unwanted data (for example, with multimedia data such as MP3 and JPEG files). Adopting this approach, the SD was left connected to the TS resulting in no clues about the automation.

The TS has been implemented as a VM in order to speed up the overall analysis process. The use of a VM allowed

the DFA to create different snapshots of the system at their will, for example, just before or after the execution of the automation procedure.

B. Analysis Methodology

The DFA had full knowledge about the methods, procedures and technologies exploited to construct the digital alibi, hence their analysis mainly focused on 4 aspects.

OPERATING SYSTEM AND APPLICATION ANALYSIS: The system structures typically containing information about executed applications (for example, the Windows Registry, the browser history, etc.) were analyzed in order to verify whether the automation produced artifacts due to its execution. Any evidence being part of the alibi (website visit, mail sending, document creation, etc.) has also been collected in order to reconstruct the time baseline.

TIMELINE ANALYSIS: This analysis focused on the identification of the files accessed or modified during the construction of the alibi in order to detect any relationships with the automation.

FILE CONTENT ANALYSIS (SD CARD): The files stored on the SD were analyzed in order to find any suspicious traces.

KEYWORD SEARCH ANALYSIS: A set of keywords collected from the source code of the automation has been used to perform a deep low-level search analysis of the SD (including allocated and unallocated space) in order to find any possible clues of the automation.

IV. CASE STUDY

In this section the strategies adopted to forge the digital alibi are presented, along with the techniques and the tools used to implement the automation and to analyze the system where it has been executed. The false digital alibi constructed by the AM is based on the digital evidence produced by the sequence of user activities summarized in Table I. Of course all the activities involving remote services are also validated by the remote providers as trusted third parties.

TABLE I
EVENT TIMELINE OF THE DIGITAL ALIBI.

Instant	Activity
t_0	Access to a Facebook account and post of a status message.
t_1	Use of Gmail to compose and send an email to several recipients.
t_2	Browsing of an user photo album with a sequential visualization of images.
t_3	Copy of the last displayed image into the clipboard.
t_4	Execution of a word processor and creation of a new document.
t_5	Write of a predefined text and paste of the previously copied image into the document.
t_6	Shutdown of the system.

The automation responsible for the creation of the alibi has been implemented on a common Personal Computer running Windows 7 Home, and is compatible with any versions of Windows, starting from Windows XP. During the challenge, the automation has been executed on the TS running Windows 7 Professional. The Facebook and Gmail accounts involved into the alibi have been previously accessed from the TS to prevent unexpected responses of the relative websites.

A. Automation Strategy

As mentioned in III-A, all the needed files have been stored on a SD memory card. In addition to the files containing the automation, some "padding" files have been also added in the same directory. Moreover, all the padding files had the same name pattern and extension (i.e., `IMG_xxxx.JPG`). At the end of the entire procedure, the automation securely removed any suspicious data (including itself). Unfortunately, some traces could persist on the disk after the deletion of a file. For example, it is possible to find name and last accessed date of removed files in some filesystem structures and Superfetch history logs. In order to make such information meaningless, the AM assigned to each automation file a name coherent with the pattern of the padding file names, thus obfuscating their real content. Adopting this strategy, it is not necessary to remove any files from the SD, as they can be overwritten "in place" (i.e., using the same physical blocks) with a different content.

In practice, a secure wiping procedure has been adopted to physically overwrite the automation files with the content of some padding files. The number of files present on the SD and their names are not modified by the automation procedure. Only the content of four files has been overwritten and their "last-modified-date" metadata are updated with new values coherent with the planned event timeline. Finally, the overwriting operations made all the file contents consistent with their extensions, leaving the data on the SD and the system traces compliant with a post-mortem analysis.

In details, the SD has been initially preloaded with:

- N files with `.JPG` extension:
  - $N-2$ were real photos (which were used as padding files),
  - one was a Java archive containing the classes needed by the automation,
  - one was a common image-viewer application suitably renamed.
- one executable file (`.EXE` extension), having the same name of the image-viewer application, but containing the automation code.
- one batch script (`.BAT` extension) to launch the automation, with a common name such as `autorun.bat`

The DFA executed the automation by launching the batch script. The automation performed all the activities described in Table I at the appropriate timings. Finally, the clean-up procedure transformed the content of the SD, according to the strategy presented before.

In particular, the SD after the automation contained:

- N files with `.JPG` extension, all representing real photos.
- one executable file (`.EXE` extension), containing the original code of the image-viewer application, but preserving the initial name.

- one batch script (`autorun.bat` extension) launching a simple slide-show of the images present on the SD.

B. Architecture and Implementation of the Automation

The false digital alibi has been coded into four software modules:

- LAUNCHER: a batch script (`autorun.bat`), which should be launched with *administrator* privileges, used to configure and to start the automation.
- SCHEDULER: a Java class (`HelloWorld.class`) stored in the JAR archive named `IMG_5719 (copy).JPG`, which provides the timed and ordered execution of the automation modules.
- SIMULATOR: an executable file, compiled in native code without external dependencies. Despite of its name (`IrfanViewPortable.exe`), it simulates all the necessary human-machine interactions to perform the activities listed in Table I.
- WIPER: a Java class (`HelloWorld1.class`) contained in the JAR file `IMG_5719 (copy).JPG`, which implements the secure wiping procedure to remove the unwanted evidence.

It is worth to note that the execution of programs from the Windows Command Prompt produces less traces in the Registry with respect to applications launched from the Windows GUI (see [4]). For this reason, the Launcher of the automation has been designed as a batch script, whose instructions are always executed from the Command Prompt. The Launcher invokes the remaining automation modules, which are in turn executed from the Command Prompt with the same benefits.

Once launched, the Scheduler prompts the user for the starting time of the alibi, then sleeps and starts the Simulator at the established time. When the simulation ends, the Wiper is invoked in order to clean-up the system.

The activities listed in Table I have been coded into the Simulator by means of a specific tool, namely AutoIt [17]. It provides an easy-to-use framework which allows to automate a sequence of systematic and repeatable human-machine interactions, such as system configuration tasks or test cases. While a number of different tools are suitable to construct an automation, AutoIt has been preferred in this challenge mainly for two reasons: (1.) it is easy-to-use but robust and reliable and (2.) it provides also a compiler to optionally produce a stand-alone executable, without external dependencies, which does not require the installation of additional software on the TS. Obviously, in the context of a digital alibi, the executable file implementing the automation is part of the unwanted evidence and must be therefore carefully removed.

The Wiper module transforms the content of the SD according to the technique described afterward. The Wiper module uses different secure deletion strategies depending on the file to be removed. In particular, the content of the file `autorun.bat` is overwritten with some non-suspicious textual data coherent with the file format and the operational environment. This strategy, named Injection Techinque, has been introduced in [12]. The content of the Launcher after the execution of the automation consists of a single line (`for /f %%f in ('dir /b *.jpg') do IrfanViewPortable.exe %%f`) which effectively launches a simple slide-show using the images stored on the SD.

The secure deletion of the Simulator (`IrfanViewPortable.exe`) is also performed by means of the Injection Technique. After its execution, the file content is completely overwritten with the content of the original implementation of the image-viewer software (IrfanView [18] portable version). This new content was initially stored on the SD card in a file named `IMG_9785.JPG`. After that the `IrfanViewPortable.exe` file has been replaced with the proper content, the source file `IMG_9785.JPG` is itself cleaned-up overwriting its data blocks with the content of the image named `IMG_9785 (copy).JPG` present on the SD.

The deletion of the JAR file containing the Scheduler and the Wiper modules (i.e., `IMG_5719 (copy).JPG`) is the most complex task because the Wiper must replace itself whilst running. As discussed in [12], a self-deletion technique must be adopted. While for native executables the OS typically locks the file to preserve the read-only semantic of the text segment, this does not occur for programs written in interpreted languages. In fact, in such a case the code is not directly executed but "translated" on-the-fly by means of an interpreter. According to this consideration, the Wiper module has been implemented in Java, which uses the Java Virtual Machine (JVM) to interpret the bytecode produced by the compiler. When started, the JVM loads all the requested classes from the JAR archive `IMG_5719 (copy).JPG` and invokes the Wiper as last step. This module can overwrite the JAR itself by using an image from the SD (`IMG_5719.JPG`) as data source. No renaming operation was performed because the JAR file has been originally named `IMG_5719 (copy).JPG`, consistently with its new content.

The Injection Technique does not guarantee that all the physical blocks of a file are overwritten. Clearly, it is necessary that the size of the new content is greater then the old content, otherwise part of the old data could be retrieved by a file carving procedure. In order to avoid this problem, the size of the files used as source of the Injection Technique has been properly chosen.

C. Automation Setup

It is fundamental to avoid unwanted information being recorded in system-protected structures that cannot be edited (and therefore cleaned) by the Wiper. The loading of the automation onto the TS is a crucial aspect for the undetectability of the entire process, as it must be accomplished without producing unwanted evidence. In these paragraphs all the preliminary precautions necessary for a clean execution of the automation are described.

1) Environment Setup: The Prefetch [19] and Superfetch [20] [21] features of the Windows Memory Manager [22] aim to speed up the applications startup by preloading their code and data into the main memory. This per-

formance improvement is accomplished by recording some ready-to-be-accessed information about used programs and files in a caching area of the main filesystem (i.e., the `C:\Windows\Prefetch` folder). However, the execution from an external drive prevents the creation of prefetch files (`C:\Windows\Prefetch\*.pf`) regarding the automation. Information about applications and data (such as filenames and access date) are logged into history files (`C:\Windows\Prefetch\*.db`) of the Superfetch [23]. However, thanks to the obfuscation techniques presented in the previous paragraphs, such information only reveals the execution of an application named `IrfanViewPortable.exe` and the access to some `JPEG` files.

In order to prevent copies of code, or data belonging to the automation, onto the disk (i.e., in the `C:\pagefile.sys` file), the amount of Virtual Memory of the system has been set to `0`.

Another feature of Windows 7 that could produce unwanted evidence is the Volume Shadow Copy Service (VSS) [24], which could accidentally create a restore point including information regarding the automation. Since this service does not perform copies of data stored on external devices by default, the presence of the automation modules on the SD produces no unwanted traces. Moreover, the use of a SD has also avoided any presence of the automation onto the main drive of the TS.

Some antivirus/antimalware software, such as Avast! [25], could detect the automation as a threat. For sake of simplicity, no application like that has been installed on the TS.

The use of a log-structured/transactional/journaled filesystem to store the automation files could determine unwanted traces due to history metadata. Even though the main drive of the TS was equipped with a journaled filesystem (NTFS) the use of an external SD to maintain the automation files with a non-journaled filesystem (FAT32) avoids such risk.

2) Automation Loading: A standard SD with capacity of 8 GB has been used for the experiment as described in the following. First of all, its content has been wiped by using Active KillDisk [26]. Afterward, the SD has been mounted on a Canon EOS 1000D camera and has been formatted by means of the embedded software, which created a FAT32 filesystem with a standard directory structure. Several pictures have been taken with the camera, resulting in a set of different JPG files getting stored on the SD. The memory has been then connected to the developing system by means of an USB adapter. A portable version of the IrfanView software (renamed to `IMG_9785.JPG`) has been copied onto it, along with the automation files `autorun.bat`, `IMG_5719 (copy).JPG` and `IrfanViewPortable.exe`), constructed as described in subsection IV-B. Thereafter the SD has been enveloped into an anti-static bag and sent by express mail to the DFA.

3) Automation Setting: Some preliminary operations have been accomplished by the DFA team before executing the automation. In particular, a SATA hard-disk has been formatted with the NTFS filesystem and mounted on the TS. Afterward, the TS has been booted from a pre-installed hard-disk containing Microsoft Windows 7 Professional. A new VM with Windows 7 Home, stored on the just mounted hard-disk, has been powered up by means of VMware Player [27].

Before running the automation, according to the setup instructions given by the AM, the DFA performed the following system customizations: (1.) the screen resolution has been set to 1024x768 pixels, (2.) Mozilla Firefox 8 and Java Runtime Environment (version 6 update 30) have been downloaded through Internet Explorer and installed, (3.) the pop-up blocker of the browser has been disabled, (4.) the Gmail and Facebook websites have been accessed several times in order to both test their reachability and create a behavioral pattern of the user (which is a sort of digital profile). Finally, the VM has been turned off and a forensic copy of it has been acquired and stored on the host OS, in order to keep a system snapshot before launching the automation.

4) Automation Execution: To create the false digital alibi, the VM has been booted up and the SD containing the automation has been connected to the system by means of an USB adapter. The DFA opened the just mounted drive (`E:`) through the Windows file manager and executed the Launcher (`autorun.bat`) with administrator privileges (`right click -> run as administrator`). It started the Scheduler, which asked the user for the starting time of the alibi and deferred the launch of the Simulator until that moment. After the automation performed all the planned actions, the system has been automatically turned-off, as expected.

D. Analysis Techniques and Tools

Having complete knowledge of the methods and techniques adopted to construct the digital alibi, the DFA analysis focused on two main elements: (1.) the SD found on the system and (2.) the VM hosting the TS where the automation has been executed.

In order to preserve the original snapshot of the TS after the execution of the automation, the VM has been backed-up and its read-only property has been preserved. A forensic copy of the SD has been acquired by means of a hardware USB Write-Blocker [28] and the `dd` tool [29]. The MD5 hash of both the SD content and the VM has been computed for further verifications. The main tools used by the DFA to perform the analysis of the TS are:

- RegRipper2 [30] to analyze the Windows Registry.
- Mozilla History View [31] and Mozilla Cache View [32] to collect any evidence left by the browser activity.
- AccessData Forensic Toolkit [33] to analyze the SD content and the main drive of the TS.
- AccessData Forensic Toolkit (DT Search engine) to perform a low-level keyword-based search operation on the involved digital memories.

In particular, the latter activity consisted of a search for some keywords considered as *signatures* of the automation modules, selected from the automation code initially stored on the SD, in the entire physical space of both the SD and the main drive of the TS. More information on file carving

techniques can be found in [34]. This operation could reveal hidden information about the automation procedure, such as metadata stored in the filesystem structure, deleted contents of files, etc. Later on, the search results have been carefully analyzed by the DFA in order to disclose any connections with the automation procedure. Some examples of keywords selected from the code of the automation are: `IMG85719`, `IrfanView`, `HelloWorld`, `AutoIt`, `WIPER`, `AUTOM`.

E. Analysis Results

This section points out the main outcomes of the analysis performed by the DFA. The analysis focused on the PC (the VM running the TS) where the automation has been executed and particularly on the SD Card used to transfer the necessary files.

SD ANALYSIS The analysis of the SD only stated that some files have been accessed or modified during the alibi timeline, in a way that is coherent with the alibi. In particular:

- The modification date of the file `IMG_5719 (copy).JPG` was changed, and its content was exactly the same of of `IMG_5719.JPG`.
- The file `autoexec.bat` was modified and its content was found to be "`for /f %%f in ('dir /b *.jpg') do IrfanViewPortable.exe %%f`".
- The modification date of the file `IrfanViewPortable.exe` has been set to a time within the alibi timeline. However, its content was exactly the original image-viewer application.
- The remaining files stored on the SD (padding files) were still present and untouched.
- The low-level keyword search operation performed on both the allocated and unallocated space of the SD did not produce any interesting results, and no reference to the automation procedure was detected.

These results show that no clue about the automation and its execution was found on the SD.

VM ANALYSIS The analysis of the VM where the automation procedure run revealed the following results:

- The traces left by the applications used during the alibi timeline (Firefox cache and history, Wordpad metadata, etc.) confirmed that all the events claimed by the AM (see Table I) were effectively performed on the TS.
- The analysis of the artifacts produced by the OS (referenced and unreferenced Registry entries, prefetching information, system logs, etc.) were coherent with all the activities performed by the user.
- The low-level keyword search operations performed on the main drive of the TS detected several locations containing the filenames used by the automation.

The filenames found with the keyword search in both the allocated and the free space of the disk did not give any clue related to the automation.

As a further confirmation, the Superfetch history files (`*.db`) stored in the `C:\Windows\Prefetch` folder of the TS have been decoded (see [23] for details) and analyzed in order to find out any suspicious program. Even though these logs typically retain a full history of the used applications and data files, the DFA team was not able to extract any meaningful information.

A deep analysis of the Registry let the DFA to reconstruct the application execution history during the alibi timeline. It is worth recalling that deleted entries can still be found in an unreferenced space of the Registry file. Its content has been recovered by means of RegRip2 in order to filter useful information about any anomalous user behavior. A snippet of the extracted data is listed below.

```
ClassGUID                                                          1
{533c5b84-ec70-11d2-9505-00c04f79deaf}                             2
Driver                                                             3
{533c5b84-ec70-11d2-9505-00c04f79deaf}\0001                        4
##?#STORAGE#VOLUMESNAPSHOT#HARDDISKVOLUMESNAPSHOT2#               5
  {53f5630d-b6bf-11d0-94f2-00a0c91efb8b}                           6
DeviceInstance                                                     7
STORAGE\VOLUMESNAPSHOT\HARDDISKVOLUMESNAPSHOT2                     8
SymbolicLink                                                       9
\System32\cmd.exe                                                  10
\Windows\system32\mode.com                                         11
\Windows\system32\java.exe                                         12
\IrfanViewPortable.exe                                             13
\Windows\System32\DeviceCenter.dll                                 14
\Windows\system32\verclsid.exe                                     15
\Windows\System32\tquery.dll                                       16
\Program Files\Mozilla Firefox\firefox.exe                         17
\Windows\system32\taskhost.exe                                     18
\Program Files\Windows NT\Accessories\wordpad.exe                  19
\Windows\System32\EhStorShell.dll                                  20
\Windows\System32\cscui.dll                                        21
\Windows\System32\ntshrui.dll                                      22
\Windows\System32\networkexplorer.dll                              23
\Windows\system32\SearchProtocolHost.exe                           24
\Windows\system32\SearchFilterHost.exe                             25
```

The lines `10`, `12` and `13` correspond to the execution, respectively, of the Launcher, the Scheduler and the Simulator modules. However, the presence of this information cannot be considered suspicious since the respective filenames have been obfuscated in the design phase.

In conclusion, the DFA did not found traces revealing that the digital alibi claimed by the AM was forged by an automation procedure.

V. CONCLUSIONS

In this work an alibi has been forged according to the methodologies and techniques presented in [4]. The authors findings are supported by a real experiment documented in this work. It consisted of a challenge between two teams: the AM, in charge of constructing the automation; the DFA, responsible for both the execution of the procedure and the analysis of the system. Despite all the useful implementation details were revealed to the DFA, no clue was found revealing that the digital alibi was produced by an automation procedure rather that by a real user. Therefore, the forged alibi has resulted to be resistant against a Digital Forensics analysis conducted on a "superset" of the evidence that could normally be collected in real cases.

When setting up a false alibi the last thing one may want to focus on is failure. Clearly, the chances of failures increase as

the alibi becomes more complex. By running the experiment, the authors have learned that there are several reasons why a false digital alibi may fail to work:

Poor planning of actions. The elements of the alibi can seem suspicious if consisting of actions not usually performed. For example, there might be too many actions or the sequence of actions might exhibit a very regular behaviour (like a constant delay between each other).

Insufficient knowledge of the system and the tools. Unexpected traces of the automation may be left on the filesystem or recorded by a system service. Only a deep knowledge of both the host OS and the automation tools allows to prevent any unwanted information. This issue demands for a continuous upgrade of the techniques since OSes are constantly evolving.

Unthought changes of external services. A service involved in the alibi may change in such a way that the automation did not consider. For example, the webpage of the email service used by the automation could be upgraded.

Unexpected events. Many different unpredictable events may undermine a false digital alibi. For example, the PC can be infected by a virus or have hardware problems such as failure of a component or power interruption. Other events, even though unlikely, may occur such as the intrusion of a burglar, or the presence of a video surveillance system which contradicts (part of) the alibi timeline.

However, it is important to highlight that if none of the above cases occurs, it is impossible to distinguish whether the digital alibi has been produced by a human interaction or by an automated program. Therefore, the results of a Digital Forensics analysis always needs to be planted in the context of the framework of all direct and circumstantial evidence.

ACKNOWLEDGEMENTS

The authors would like to thank many friends from IISFA (International Information System Forensics Association) and in particular Francesco Cajani (Deputy Public Prosecutor High Tech Crime Unit, Court of Law in Milano) and Mario Ianulardo (Computer Crime Lawyer in Napoli) for their many interesting and stimulating discussions.

A special thank goes to Litiano Piccin for his precious support during the preliminary challenge held for the OLAF 2011 event.

REFERENCES

[1] Miniwatts Marketing Group, "Internet World Stats," http://www.internetworldstats.com/stats.htm, December 31, 2011.

[2] F. Palmieri, U. Fiore, and A. Castiglione, "Automatic security assessment for next generation wireless mobile networks," *Mobile Information Systems*, vol. 7, no. 3, pp. 217–239, 2011.

[3] U.S. Legal, Inc., "Legal Definitions and Legal Terms Dictionary," http://definitions.uslegal.com/.

[4] A. De Santis, A. Castiglione, G. Cattaneo, G. De Maio, and M. Ianulardo, "Automated Construction of a False Digital Alibi," in *MURPBES*, ser. Lecture Notes in Computer Science, A. M. Tjoa, G. Quirchmayr, I. You, and L. Xu, Eds., vol. 6908. Springer, 2011, pp. 359–373.

[5] Merriam-Webster Inc., "Online dictionary," http://www.merriam-webster.com/.

[6] F. Palmieri and U. Fiore, "Network anomaly detection through nonlinear analysis," *Computers & Security*, vol. 29, no. 7, pp. 737–755, 2010.

[7] P. Albano, A. Castiglione, G. Cattaneo, G. De Maio, and A. De Santis, "On the Construction of a False Digital Alibi on the Android OS," in *INCoS*, F. Xhafa, L. Barolli, and M. Köppen, Eds. IEEE Computer Society, 2011, pp. 685–690.

[8] P. Albano, A. Castiglione, G. Cattaneo, and A. De Santis, "A Novel Anti-forensics Technique for the Android OS." Los Alamitos, CA, USA: IEEE Computer Society, 2011, pp. 380–385.

[9] P. Gutmann, "Secure Deletion of Data from Magnetic and Solid-State Memory," in *Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography - Volume 6*. Berkeley, CA, USA: USENIX Association, 1996, pp. 77–89.

[10] P. Gutmann, "Data Remanence in Semiconductor Devices," in *Proceedings of the 10th conference on USENIX Security Symposium - Volume 10*. Berkeley, CA, USA: USENIX Association, 2001, pp. 4–4.

[11] U.S. Department of Defense, *DoD Directive 5220.22, National Industrial Security Program (NISP)*, http://www.dtic.mil/whs/directives/corres/html/522022m.htm, February 2010.

[12] A. Castiglione, G. Cattaneo, G. De Maio, and A. De Santis, "Automatic, Selective and Secure Deletion of Digital Evidence."

[13] European Lawyers' Union, "Transmission of Evidence in Criminal within the European Union and Protection of the European Union's Financial Interests," http://www.uae.lu/system/doc/UAE_019/UAE-OLAF-JANVIER-2011-Programme.pdf, January 2011.

[14] IISFA, "International Information Systems Forensics Association," http://www.iisfa.net/, (accessed January 2012).

[15] M. Epifani, "Alla Caccia del Falso Alibi? (in Italian)," *Symposium - Transmission of evidence in criminal within the European Union and protection of the European Union's financial interests*, (accessed January 2012), http://www.iisfa.net/index.php?option=com_docman&task=doc_download&gid=119&Itemid=23.

[16] K. Lee, H. Yeuk, Y. Choi, S. Pho, I. You, and K. Yim, "Safe Authentication Protocol for Secure USB Memories," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 1, no. 1, pp. 46–55, June 2010.

[17] J. Bennett, "AutoIt v3.3.6.1," http://www.autoitscript.com/autoit3/, (accessed January 2012).

[18] I. Skiljan, "Irfanview," http://www.irfanview.com/, January 2012.

[19] M. Russinovich, *Microsoft Windows internals: Microsoft Windows Server 2003, Windows XP, and Windows 2000*. Redmond, Wash: Microsoft Press, 2005.

[20] T. Holwerda, "SuperFetch: How it Works & Myths," http://www.osnews.com/story/21471/SuperFetch_How_it_Works_Myths, May 2009.

[21] M. Russinovich, "Inside the Windows Vista Kernel: Part 2," http://technet.microsoft.com/en-us/magazine/2007.03.vistakernel.aspx, March 2007.

[22] Microsoft MSDN, "Memory Management," http://msdn.microsoft.com/en-us/library/windows/desktop/aa366779(v=vs.85).aspx, (accessed January 2012).

[23] ReWolf, "Windows SuperFetch file format — partial specification," ReWolf's blog — http://blog.rewolf.pl/blog/?p=214, October 2011.

[24] Microsoft MSDN, "Volume Shadow Copy Service," http://msdn.microsoft.com/en-us/library/windows/desktop/bb968832(v=vs.85).aspx, July 2011.

[25] AVAST Software a.s., "avast!" http://www.avast.com/, 2012.

[26] LSoft Technologies Inc., "Active KillDisk," http://www.killdisk.com/, 2011.

[27] VMware, Inc., "VMware Player," http://www.vmware.com/products/player/, 2012.

[28] WiebeTech, "Forensic in-line USB WriteBlocker," http://www.wiebetech.com/products/USB-WriteBlocker.php, 2011.

[29] GNU Project, "GNU Coreutils — dd invocation," http://www.gnu.org/software/coreutils/manual/html_node/dd-invocation.html, Accessed January 2012.

[30] H. Carvey, "RegRipper," http://regripper.wordpress.com/, February 2012.

[31] N. Sofer, "Mozilla History View v1.43," http://www.nirsoft.net/utils/mozilla_history_view.html, 2012.

[32] N. Sofer, "Mozilla Cache View v1.51," http://www.nirsoft.net/utils/mozilla_cache_viewer.html, 2012.

[33] AccessData Group, LLC, "Forensic Toolkit v4," http://accessdata.com/products/computer-forensics/ftk, 2011.

[34] R. Poisel, S. Tjoa, and P. Tavolato, "Advanced file carving approaches for multimedia files," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 2, no. 4, pp. 42–58, December 2011.