MIAT-WM5: FORENSIC ACQUISITION FOR WINDOWS MOBILE POCKETPC

Fabio Dellutri, Vittorio Ottaviani, Gianluigi Me

Dip. di Informatica, Sistemi e Produzione - Università degli Studi di Roma "Tor Vergata"
Via del Politecnico 1, 00133 Rome, Italy
Email: {dellutri,ottaviani,me}@disp.uniroma2.it

KEYWORDS

Mobile Forensics, Data Seizure, PDA, PocketPC, Windows Mobile

ABSTRACT

A PocketPC equipped with phone capabilities could be seen as an advanced smartphone, providing more computational power and available resources. Even though several technologies have emerged for PDAs and Smartphones forensic acquisition and analysis, only few technologies and products are capable of performing forensic acquisition on PocketPC platform; moreover they rely on proprietary protocols, proprietary cable-jack and proprietary operating systems. This paper presents the Mobile Internal Acquisition Tool for PocketPC devices. The approach we propose in this paper focuses on acquiring data from a mobile device's internal storage memory, copying data to an external removable memory (like SD, mini SD, etc.). Such task is performed without the need of connecting the device to PC. Thanks to this, forensic operators could avoid to travel with luggage plenty of one-on-one tools for every single mobile device. Finally, we will show some experimental results, comparing this methodology with standard products on real world devices.

INTRODUCTION

The mobile phone can be considered the ultimate disruptive technology: in fact, like telephony, radio, television, and the Internet, mobile phones are dramatically changing nearly every aspect of daily life, both inside businesses and in the daily lives of individuals, providing more applications and collecting more private data. The enriched capabilities of new smartphones (118 million in 2007, Canalys), such as multi-connectivity (HDSPA, Bluetooth, IR, WLAN) and multimedia recording, make the content of the mobile device memory very interesting from a criminal investigation perspective. In particular, the growing prominence of forensic sciences, in the investigation chain, led to a broad use of forensic tools to acquire mobile phone memory content, to witness the evidence of a crime. However, as rule of thumb, the crime-scene usually offers many different mobile phone/smartphone models, causing the forensic operators to be overwhelmed by using the one-on-one connectors for every single mobile device. In fact, current acquiring tools, adopted by forensic operators, extract the internal memory remotely (typically via USB), with proprietary mobile phone connectors: a forensic tool running on a laptop is connected with the target device and, using the OS services, it extracts the data like SMS, MMS, TODO list, pictures, ring tones etc. This approach has the advantage

- to minimize the interaction with the device and
- to automate the procedure of the interpretation of the seized data.

However, the main disadvantage is the partial access of the file system, which relies on the communication protocol. As discussed above, since many protocols are proprietary, we can not see how many effects the data exchanged have on the memory status. For this reason, we have developed a tool to acquire mobile phone memory by a SD/MMC (Secure Digital / MultiMediaCard) memory inserted in the available mobile phone SD/MMC slot, called MIAT (Mobile Internal Acquisition Tool) for Symbian (Me et al., 2008). In the fourth quarter of 2007, Canalys estimated that Symbian had a 65% share of worldwide converged device shipments, ahead of Microsoft on 12% and RIM on 11% (Canalys, 2007). In this paper we present the MIAT for the PocketPC platform running Windows Mobile (MIAT-WM5), which cannot be considered, roughly, as a porting because of the differences between Symbian and Windows Mobile operating systems. For this reason, the MIAT-WM5 takes advantage of Windows Operating system characteristics (e.g. filesystem, memory management etc) to maximize the outcome of the acquisition phase for Windows devices.

STATE OF ART

The term *PocketPC* refers to a Microsoft specification that sets various hardware and software requirements for a handheld-sized computer (*PDA*, Personal Digital Assistant) that runs the Windows Mobile operating system (Wikipedia, Pocket PC definition). As reported in Table 1, many tools perform forensic operations on a PDA. However, as Ayers *et al.* assert in (Ayers et al., 2007, 2004, 2005), the only NIST certified tool on a PocketPC is Paraben's *PDA Seizure*. Such tool performs data seizure of internal memory in a remote way. Actually, the

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

Table 1: PDA forensic tools

	Palm OS	PocketPC	Linux PDA
pdd	Acquisition	NA	NA
Pilot-Link	Acquisition	NA	NA
PDA Seizure	Acquisition, Examination, Reporting	Acquisition, Examination, Reporting	NA
EnCase	Acquisition, Examination, Reporting	NA	Examination, Reporting
POSE	Examination, Reporting	NA	NA
dd	NA	NA	Acquisition

forensic tool is connected to a device by cradle or USB cable and, through the Microsoft's ActiveSync protocol, it extracts data such as user's files, call logs, SMS, MMS, TODO list, etc. This approach has the advantage to minimize the interaction towards the device and to automate the process of seized data interpretation. The main disadvantage relies on the protocol closeness: we are not able to measure any memory alteration caused by data exchange. Moreover, to perform the acquisition process, PDA seizure degrades the evidence putting a dll file in the device's file system.

POCKETPC INTERNAL MEMORY AND STORAGE ARCHITECTURE

In Windows Mobile 2003 PocketPC and earlier, device's memory was split in two sections: a ROM section, containing all operating system core files, and a RAM section aimed in keeping the user storage (Storage Memory) and the memory space for running applications and their data (Program Memory). The user can choose the amount of memory to be reserved to Storage Memory and then to the Program Memory. The RAM chip was built on a volatile memory scheme, so a backup battery was required to keep the RAM circuitry powered up, even if the device was just suspended. In case battery power supply went down, all user's data were lost. Such scenario forced user to recharge battery within a time limit of 72 hours (as mandatory by Microsoft to devices manufacturers).

Since Windows Mobile 5, memory architecture was redesigned to implement a non-volatile user storage.

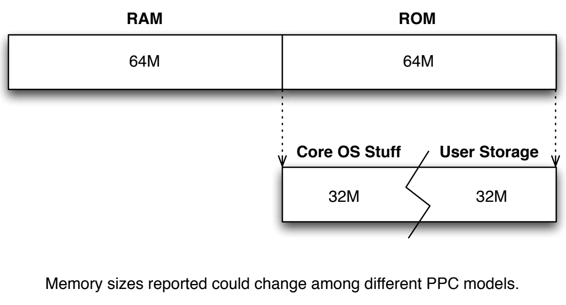

RAM
ROM
64M
64M
Core OS Stuff
User Storage
32M
32M
Memory sizes reported could change among different PPC models.

Figure 1: Windows Moble 5.0 memory architecture.

Currently, the memory is split in two section (see Figure 1): the RAM is aimed to hold running processes data, whereas the ROM keeps core OS code and libraries (called modules), the registry, databases and user's files. Such memory, also called Persistent Storage and contained within a flash memory chip, can be built using many different technologies (Santarini, 2005):

- **XIP model**, based on NOR memory and volatile memory, this technology enables device to store modules and executables in XIP (execute-in-place) format and allows the operating system to run applications directly from ROM, avoiding to copy them first in the RAM section. NOR memory has poor write performance.

- **Shadow model**, which boots the system from NOR and uses a NAND for the storage. This model is power-expensive, because the volatile memory requires to be constantly powered on.

- **NAND store and download model**, which reduces costs replacing NOR with OTP (one-time programmable) memory model.

- **Hybrid store and download model**, which mixes SRAM and NAND, covering them with a NOR-like access interface (to support XIP model).

Windows Mobile 5 and above place the great part of the applications and system data in the Persistent Storage. Core OS files, user's files, databases and registry are seen by applications and users in the same file system tree, which is hold and controlled by the FileSys.exe process. Such process is also responsible for handling the Object Store, which maps objects like databases, registry and user's files in a contiguous heap space. The Object Store's role is to manage the stack and the heap memory, to compress and to expand files, to integrate ROM-based applications and RAM-based data. For a comprehensive explanation about how Windows Mobile uses the Object Store and manages linear flash memory, see (Microsoft, Linear Flash Memory Devices on Microsoft Windows CE.) and (Microsoft, The Windows CE 5.0 Object Store.).

The strategy for storing data is based on a transactional model, which ensures that store is never corrupted after a power down while data is being written. Finally, the Storage Manager manages storage devices and their file systems, offering a high-level layer over storage drivers, partition drivers, file system drivers and file system filters.

OUR METHODOLOGY

The approach we propose in this paper focuses on acquiring data from a mobile device's internal storage memory, copying data to an external removable memory (like SD, mini SD, etc.). Such task is performed without the need

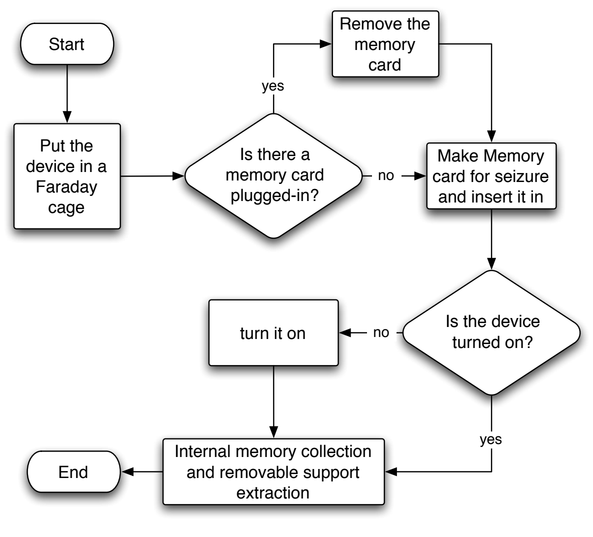

Start
Put the device in a Faraday cage
Is there a memory card plugged-in?
yes
Remove the memory card
no
Make Memory card for seizure and insert it in
Is the device turned on?
no
turn it on
yes
Internal memory collection and removable support extraction
End

Figure 2: Data collection workflow

of connecting the device to PC. Thanks to this, forensic operators could avoid to travel with luggage plenty of one-on-one tools for every single mobile device.

The complete data seizure process is shown in Figure 2. In order to acquire the memory content of a GSM, Bluetooth or Wi-Fi enabled mobile device, it is mandatory to shield the device with a Faraday cage (Leyland, 1992). Indeed, new incoming calls, SMS, e-mails, Bluetooth activity, connection status changes or GSM cell switch, could trigger events which may modify some file system's objects. Unlike old Symbian smartphones, where we were forced to remove battery supply to remove the memory card, in a standard PocketPC it is possible to plug-in a memory card (typically an SD) while the device is powered-on (*hotplug*). This is a great chance for collecting data which, otherwise, could be altered if the device was turned off before the seizure process. Therefore, we have to check first if a memory card is already plugged, and replace it with a memory card containing MIAT-WM5. Moreover, if the device is turned off, now it can be started. MIAT-WM5 can be set as autorunnable, to avoid to start applications like fexplore.exe to navigate through the filesystem and to launch the seizure application; indeed it is important to run as few applications as we can, to avoid locking problems and changes in the file system triggered by other processes. Anyway, MIAT-WM5 kills all non-necessary processes running on the system in order to avoid lock problems. MIAT-WM5 performs a hashing of each file before and after the copy, to ensure acquired image integrity. The report containing file hashes is saved in a log file. Data stored in the original memory card can be acquired using a MMC or SD reader (USB or integrated) and a byte stream imaging tool (like DD): binary data are read from source, then stored as an image file, representing all the single bytes, including file system's metadata. After that, it is possible to analyse the file allocation table to recover deleted data. When internal memory seizure has been done, SIM card could be removed and analysed with specific tools (Oxygen Software, Forensic; Casadei et al., 2005).

IMPLEMENTATION DETAILS

We have chosen to develop the application using a native C++ approach, fulfilling the requirement of having a tool to be launched from an external memory card, without the need of a pre-installed runtime environment (like java virtual machine), neither the need to install the tool on the device. The application runs in stand-alone mode, and it does not require any third party's dll. Since the tool uses the standard Windows Mobile APIs to access the file system (like Open, Read and Write, FileCopy), we can reasonably think that these APIs will not change in future versions of OS: then the forward compatibility can be assured. In Algorithm 1 is depicted the pseudo-code of the seizure process, that starts after the main application killed all the other non-vital running processes.

Algorithm 1 Seizure

```
Input: A path p.
Output: none.

for all objects obj (files and directories) in p do
  if obj is a directory then
    Create a directory named p in the SD Card
    Recursively call Seizure(p/obj)
  else if obj is a file then
    Compute MD5 hash of obj
    Copy obj in path p on the SD Card
    if obj has not been copied then
      Access to obj with CEDB APIs
      if obj could be accessed then
        recreate a similar database in path p on the SD Card
      end if
    end if
    Compute MD5 hash of the copied obj on the SD Card
  end if
end for
```

Such algorithm performs two main tasks:

- the *copy* task, which copies all internal memory's files of the mobile device on the memory card;
- the *hash* task, which ensures the integrity of the copied files and allows to discover which files have been modified during the seizure process.

The *Seizure* algorithm works using APIs like CopyFile Open Close, and it copies recursively every internal file system entry on the memory card. This task preserves the directory structure, copying files according to their original position. The hash task computes the MD5 hash of each file found in the device internal memory. Hashes are written in a log file saved in a separate directory. The

Table 3: MIAT-WM5 and PDA seizure comparison, and their hashes consistency

File	Paraben	MIAT
/Documents And Settings/default.vol	–	⋆
/Documents And Settings/system.hv	–	–
/Documents And Settings/default/user.hv	–	–
/Windows/*.dll	–	–
/mxip_notify.vol	√	⋆
/cemail.vol	√	⋆
/mxip_system.vol	√	√
/mxip_lang.vol	√	√
/pim.vol	⋆	√

– file not copied
⋆ file copied but its hash does not match
√ file copied and hash matches

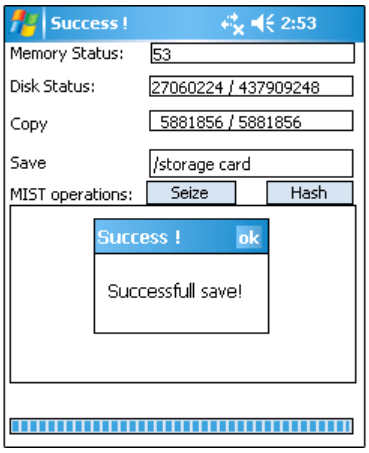
hash task can be launched as a separate function, and it surfs the whole filesystem to compute hash of every files.

The *Seizure* algorithm invokes the hash function before and after the copy of every single file, allowing to understand if changes happen during the copy from the internal filesystem to the Storage Card.

As reported in Section "POCKETPC INTERNAL MEMORY AND STORAGE ARCHITECTURE", Windows Mobile places OS's stuff in a lot of file-like objects in the same file system seen by the user (under `/Windows` directory). Most of these files are inaccessible by the standard file system APIs because they are objects that are in XIP format: most of the headers are removed and the addresses are fixed up so that the programs are able to run with no need to be loaded into RAM first. The binary has been stripped down and customized for that particular device (Yost, 2007). Such files are also flagged with file attributes like `FILE_ATTRIBUTE_INROM` and `FILE_ATTRIBUTE_ROMMODULE`. Our application skips these files: there is no reason to look for a method to access such files because they are firmware's modules and they could be replaced with new ones only by an advanced user (using the *ROM flashing* technique - e.g. if she is willing to upgrade her firmware with a new version of the operating system or she want to modify things like bootsplash). Moreover, there is another set of files that cannot be accessed by standard APIs: these files are database objects locked by operating system processes which cannot be killed. We reach to access their data using CEDB APIs and we are able to recreate such files in the external memory card. In Table 2 it is shown where most relevant data about user and system are stored in the file system.

EXPERIMENTAL RESULTS

The working approach of MIAT-WM5 and Paraben is quite different. MIAT-WM5 scans the filesystem of the PocketPC saving data in the external memory card, as Paraben seems to get data from ROM memory at a lower level than MIAT-WM5. There are some differences between Paraben and MIAT-WM5, first of all Paraben


Success ! 2:53

Memory Status: 53
Disk Status: 27060224 / 437909248
Copy 5881856 / 5881856
Save /storage card
MIST operations: Seize Hash

Success ! ok
Successfull save!

Figure 3: MIAT-WM5 screenshot after run a data seizure

needs a computer to seize data from a mobile device, as it is a Windows application. As described above, MIAT-WM5 does not need any other device to run: the operator needs only an external memory card pluggable in the device. Looking at the results of some tests done on a physical HTC device and on a emulated one (on a Windows XP computer), Paraben and MIAT-WM5 seems to extract the same files. Some differences rely on hashes of some files: Paraben seems to modify some files, such as pim.vol, and it preserves others, but, as its source code is not available, we can not explain what happens. MIAT-WM5 modifies some files, because it access their data through database APIs and writes a new file containing same data, with a resulting hash different from the original (see Table 3). As described before, OS's core files are impossible to be extracted both for MIAT-WM5 and Paraben. Another relevant consideration to be done is that Paraben reach to recover something from deleted files (just erased before the seizure), but in all the experiments these files looked like zero-padded. That suggests that Windows Mobile replaces erased block sectors very quickly, probably because the memory size is too short to preserve them for the entire seizure time. In the testing phase, we used a PC AMD Athlon64 X2 Dual 1GB Ram and a QTEK9000 PDA (HTC Universal), equipped with a Kingston SD 2GB. On such hardware, seizure times between the two solutions are quite the same: Paraben's time depends only on how many files resides on the internal storage; MIAT-WM5's time relies both on files amount and on external memory access time.

Thanks to its hashing characteristics, MIAT-WM5 was also useful to enumerate all file system's changes when device sustains events like simple reset, on-line/off-line mode change or SIM card removal. Such results are shown in Table 4. Some files' hashes are impossible to be computed ("-" symbol) because those files are locked and the system does not allow one to perform read operations on such files.

Table 2: Windows Mobile 5.0 relevant files

Filename	Location	Description
System.hv	/Documents And Settings/system.hv	System registry hive.
User.hv	/Documents And Settings/default/user.hv	User registry hive for default user.
Default.vol	/Documents And Settings/default.vol	Object store replacement volume for persistent CEDB databases. This file contains MSN contacts
Mxip_system.vol, Mxip_lang.vol, Mxip_notify.vol, Mxip_initdb.vol	/	Metabase volumes, including language-specific data and storage for notifications.
Cemail.vol	/	Default SMS and e-mail storage.
Pim.vol	/	Personal Information Manager (PIM) data, such as address book, schedules, SIM entries, call logs.

Table 4: File system changes before and after a event and files lock status

File		Before	After
Device reboot			
/Windows/VSDApp.bin	⊙	□	□
/History.txt	⊙	□	□
/mxip_system.vol	-	□	□
/mxip_lang.vol	-	■	□
From on-line to off-line mode			
/Windows/Profiles/guest/Temporary Internet Files/ Content.IE5	-	□	■
/Windows/Profiles/guest/Cookies/index.dat	-	□	■
/Windows/Profiles/guest/History/History.IE5/ index.dat	-	□	■
SIM card removal			
/mxip_system.vol	-	■	□
/mxip_lang.vol	-	■	□
/History.txt	⊙	□	□
/Windows/VSDSIMInfor.bin	⊗	□	⊗
/Windows/VSDApp.bin	⊙	□	□
/Windows/Profiles/guest/Cookies/index.dat	-	■	□
/Windows/Profiles/guest/History/History.IE5/ index.dat	-	■	□
/Windows/Profiles/guest/Temporary Internet Files/Content.IE5	-	■	□
/Windows/Start Menu/Programmi/ SIM i.TIM.lnk	⊗	□	⊗

⊙ Modified in the transition
⊗ Disappear after the change
□ Unlocked
■ Locked
- Unavailable

CONCLUSIONS

In this paper we proposed an alternative methodology to seize data from PocketPC devices, based on our belief that forensic model should be opened and verifiable. Therefore, MIAT-WM5 could be proposed as an open-source software, in order to give it transparency and verifiability.

Currently we are working on combining files copy with a full internal memory dump. With this approach, we will ensure to extract a complete and consistent snapshot of the system. This approach could involve a device-dependent code because each manufacturer uses a different memory technology (as discussed above in Section "POCKETPC INTERNAL MEMORY AND STORAGE ARCHITECTURE"), therefore it implements its own low level storage SDK (needed to find the storage addresses range into the ROM).

Moreover, we are improving the application design to support Windows Mobile 6.0 as well.

ACKNOWLEDGEMENTS

We wish to express our thanks to Giuseppe F. Italiano, who reviewed the draft of this document.

REFERENCES

Me, G., Rossi, M. (2008). *Internal forensic acquisition for mobile equipments.* 4th Int'l Workshop on Security in Systems and Networks (SSN2008), Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), 2008, IEEE Computer Society Press, TO APPEAR.

Canalys.com (2007). *Smart mobile device shipments hit 118 million in 2007, up 53% on 2006.* http://www.canalys.com/pr/2008/r2008021.htm.

Wikipedia. *Pocket PC definition* http://en.wikipedia.org/wiki/Pocket_PC

Ayers, R, Jansen, W, Moenner, L, Delaitre, A. (2007). *Cell Phone Forensic Tools: An Overview and Analysis Update.* NISTIR 7387, March 2007

Ayers, R, Jansen, W. (2004). *PDA Forensic Tools: An Overview and Analysis.* NISTIR 7100, August 2004

Ayers, R, Jansen, W. (2005). *An overview and Analysis of PDA Forensics Tool.* The International Journal of Digital Evidence, 2005

Paraben's Forensics Software *Paraben Device Seizure.* www.paraben.com

Santarini, M. (2005) *NAND versus NOR.* EDN, October 13, 2005

Microsoft. *File System Boot Process.* MSDN, msdn2.microsoft.com/en-us/library/aa912276.aspx

Microsoft. *Linear Flash Memory Devices on Microsoft Windows CE.* Microsoft TechNet, www.microsoft.com/technet/archive/wce/plan/flashce.mspx

Microsoft. *The Windows CE 5.0 Object Store* MSDN, msdn2.microsoft.com/en-us/library/ms885891.aspx

Leyland, W. J. (1992). *Lightweight portable EMI shielding container.* United States Secretaty, The Navy US. OF. (US) 5136119 http://www.freepatentsonline.com/5136119.html

Oxygen Software. www.opm-2.com/forensic/

Casadei, F, Savoldi, A, Gubian, P. (2005). *SIMbrush: an Open Source Tool for GSM and UMTS Forensic Analysis.* In proceedings of Systematic Approaches to Digital Forensic Engineering (IEEE SADFE 2005), pgg. 105-119, Taipei, TW, 7-9 November 2005.

Yost, S. (2007). *Why can't I copy programs out of Windows?.* http://blogs.msdn.com/windowsmobile/archive/2007/12/29/why-can-t-i-copy-programs-out-of-windows.aspx

AUTHOR BIOGRAPHIES

FABIO DELLUTRI is a Ph. D. student in computer science engineering at the Computer Science Engineering Department, Università degli Studi di Roma "Tor Vegata". His interests include Web applications and OS security, mobile digital forensics and experimental algorythms. His email is `dellutri@disp.uniroma2.it`.

VITTORIO OTTAVIANI is a Ph. D. student in computer science engineering at the Computer Science Engineering Department, Università degli Studi di Roma "Tor Vergata". His interests include web applications mobile and OS security, mobile digital forensics and GIS applications. His email is `ottaviani@disp.uniroma2.it`.

GIANLUIGI ME, Ph. D., is an adjunct professor of Computer Security at the Università degli Studi di Roma "Tor Vergata". He holds a strong experience in managing training for law enforcement high tech crime units and serves as a member of the advisory board of the International Journal of Electronic Security and Digital Forensics. His research interests include mobile computing applications, digital forensics, electronic/mobile payments, and game theory. He is a professional member of the IEEE. His email is `me@disp.uniroma2.it`.