Peer-to-Peer Netw. Appl.
DOI 10.1007/s12083-013-0217-3

A study on memory dump analysis based on digital forensic tools

Jungtaek Seo · Seokjun Lee · Taeshik Shon

Received: 5 January 2013 / Accepted: 21 May 2013
© Springer Science+Business Media New York 2013

Abstract The application of IT in all industrial facilities has led to the use of special-purpose systems in diverse areas. As such, special-purpose systems have increasingly become the target or path of hacking attacks. From a digital forensics viewpoint, these systems can be used to gather evidence from all the relevant digital devices such as whole systems or storage units at the scene of a crime. Notably, In case of special-purpose embedded system, unlike a conventional computing system, is almost always 'powered on' like server, the accumulated data can remain in the volatile memory. This paper focuses on analyzing ways of gathering physical memory data for application in an embedded system and of developing a test system to analyze the physical memory for verification.

Keywords Memory · Memory dump · Dump · Linux · Ram · Embedded · Forensic

1 Introduction

As IT is now applied almost ubiquitously in people's everyday lives as well as in the great majority of enterprises and industries, various special-purpose systems are now in widespread use. They are installed in the mobile phones, game boxes, automobiles and smart home appliances with which we are very familiar, as well as in industrial facilities with specific purposes such as traffic control and power generation. As the application of special-purpose systems has become ever more diverse, various types of cyber-attack that specifically target that systems or which use them as the path to a specific network are expected.

This paper focuses on digital forensic measures for investigating cyber-crimes against embedded systems, rather than defense measures against cyber-attacks. Compared to studies of general digital forensics, there have been relatively few studies of digital forensics and embedded systems, given the importance of these widely applied embedded systems. Since an embedded system is almost always 'powered on', unlike conventional computer systems, many data may remain in the main memory unit (RAM). Therefore, gathering the data accumulated in the main memory unit may be more useful than accessing auxiliary memory units like a hard disk or flash memory when gathering evidence on a case. However, existing studies of digital forensics that mostly focus on auxiliary memory units are somewhat limited in to be utilized to the main memory unit.

This paper describes the results of the investigation and the testing of measures for gathering and analyzing key data such as the temporary use input values that may remain in the RAM of an embedded system, key data that has not been encrypted, the code of an encoded program code that has been decrypted for execution, and the data of a process terminated by the user.

This paper is organized as follows: In Chapter 2, it shows related works, briefly explains memory dump and digital forensics, describes the categories of memory dump, then covers the recent trend; Chapter 3 describes actual memory dump execution and analysis; Chapter 4 describes the test used to verify that significant information can be obtained through memory acquisition and analysis; and the last chapter presents the conclusion and a possible direction for future studies.

2 Background and trend

2.1 Related works of memory dump analysis

There are some papers about memory dump analysis with various computing devices. <*Forensic analysis of the Windows*

J. Seo
The Attached Institute of ETRI, Daejeon, Korea
e-mail: seojt@ensec.re.kr

S. Lee · T. Shon (✉)
Ajou University, Suwon, South Korea
e-mail: 743zh2k@gmail.com

S. Lee
e-mail: wadeco@ajou.ac.kr

Published online: 21 June 2013

registry in memory> by Gabrriela Limon Garcia covers some tools and techniques for memory acquisition and data analysis [1]. But the tools and techniques of this paper only deal with windows-based system. Similarly, <*A survey of main memory acquisition and analysis techniques for the windows operating system*> by Stefan Vomel illustrated the prevalent concepts for creating a memory snapshot of running system. It shows some method to analysis of memory data for Windows-based system [2]. The third paper named <*FATKit: A framework for the extraction and analysis of digital forensic data from volatile system memory*> by Nick L. Petroni Jr. implement the Forensic Analysis Toolkit for volatile memory forensic analysis [3]. It can extract the tasks information in memory image file acquired from Windows-based and Linux-based system. But it needs memory mapping file from OS.

In Republic of Korea, the paper named < *The Windows Physical Memory Dump Explorer for Live Forensics*> by Jisung Han, Sangjin Lee, they make a tool for get live data in windows physical memory dump file [4]. There is another paper named <*A Study of Memory Information Collection and Analysis in a view of Digital Forensic in Window System*> by Lee S.H., Kim H.S., Lee S.J., and Lim, J.I., in this paper, authors propose a process for memory acquisition in forensic process [5].

In this paper, we explain method to make memory dump file both Windows-based and Linux-based system. Then, we analyze memory dump file to get OS information and acquire some system information from Linux-based special-purpose system without other files except memory dump file. So, when someone have to analyze the target system with only memory dump file. Our approach helps to get some information from dump file.

2.2 Background about memory dump

In a computer, data are stored in either its main memory unit or its auxiliary memory unit. RAM (Random Access Memory) is the main memory unit which retrieves the programs or data from the auxiliary memory unit and temporarily stores the information until the power is turned off. It is also called the physical memory. There are three reasons why gathering and analyzing the data contained in this physical memory is important during the step involving digital information collection. First, the physical memory contains data related to the real-time system operating environment, such as the currently-mounted file system and the list of processes being operated. Second, even the encrypted data are generally decrypted into plain statements when they are stored in the physical memory. Third, it conforms to the characteristics of embedded systems. Since an embedded system is rarely turned off, the data contained in the physical memory are not often volatized. Therefore, significant information can be obtained if analysis thereof is performed effectively.

Analyses of the physical memory are generally conducted in two steps: the physical memory dump step and the analysis step. 'Memory dump' refers to the imaging of the data contained in a physical memory 'as is', whereas 'memory analysis' means analyzing the imaged memory dump file either by using an analysis tool or by obtaining the meaningful data in the dump file using various editors. Although the active memory can also be analyzed, it is rarely used in digital forensics because the data change continuously due to the nature of the memory.

2.3 Categories of memory dump

In this part, we compare some most named dump methods. Also, we categorized three dump types by dump target. Additionally, we summarized the strengths and weaknesses of both dump methods and types.

2.3.1 Types of memory dump

There are many methods to get memory dump. We choose some of major methods and summarized the Strengths and Weaknesses in Table 1. Chosen dump methods are Tribble [6], dump using firewire, WinDD, MDD, Vmware, Hibernation, etc. Dump methods can be categorized by some characters of each methods. In case of Tribble and Firewire Dump, these methods are memory dump using hardware approach. Another methods like WinDD(32bit or 64bit), MDD, dd are software for memory dump. However, there are two more methods. Dump with VMware is the way to dump when the system operating by virtualized system.

Each method has strengths and weaknesses. For example, dump using hardware can get almost pure memory data. But it must be set the PC before examine. On the other hand, dump using software it is easy way to get memory dump, but it may occur some impact on the memory. Virtualization and Hibernation method also has their characteristics.

2.3.2 Categorization by dump type

Memory dump types can categorize by make system crash or not. In Table 2, Memory dump tools with system crash can make whole memory dump or kernel memory dump or mini memory dump. There are two big types of Non-crash dump. One is Physical memory dump, other is dump memory of only the target process.

2.4 Current trend of memory dump

2.4.1 WinDD (Windows OS)

WinDD is a tool that was developed to dump the physical memory and is distributed free of charge. Currently, it is

Table 1 Comparison of memory dump methods

Type	Method	Strengths	Weaknesses
Hardware	Tribble	There is little impact on the system since no software is installed.	The equipment has to have been installed. Since it is mostly used for research and has not yet been commercialized, its technical reliability remains to be proven.
	Firewire (IEEE 1394)	Dump is enabled even in OS, which blocks access to the memory.	The equipment has to have been installed. OS and driver attributes affect dump performance.
Software	Win32DD Win64DD	Data can be dumped regardless of the Windows version of bits.	Obtaining pure memory dump is difficult.
	MDD (Windows)	Operated at the user level.	Collection of more than 4 GB of RAM data is not possible.
	dd (Linux)	The Linux-based large volume reproduction tool can be used for storing the main memory.	Access privilege must be set in advance during kernel compiling in order to gain access to the memory file in Linux.
Virtualization	VMware Virtualization	Dump after terminating the session	A virtualization engine is needed.
Power saving mode	Hibernation	No additional program or equipment is required.	Dump only of the areas in use, not the whole memory.

distributed along with other programs as part of the Windows memory toolkit. It is available in the most popular Windows XP and Windows 7 programs. Win32DD is used in a 32-bit environment, while Win64DD is used in a 64-bit environment. WinDD reads '\Device\PhysicalMemory', which is a memory drive managed by the Windows kernel, and saves it into a file. Figure 1 below shows the result of a 2 GB memory dump with the command 'win64dd.exe –f c:\memdump.win32dd'.

2.4.2 Process dump using task manager (Windows OS)

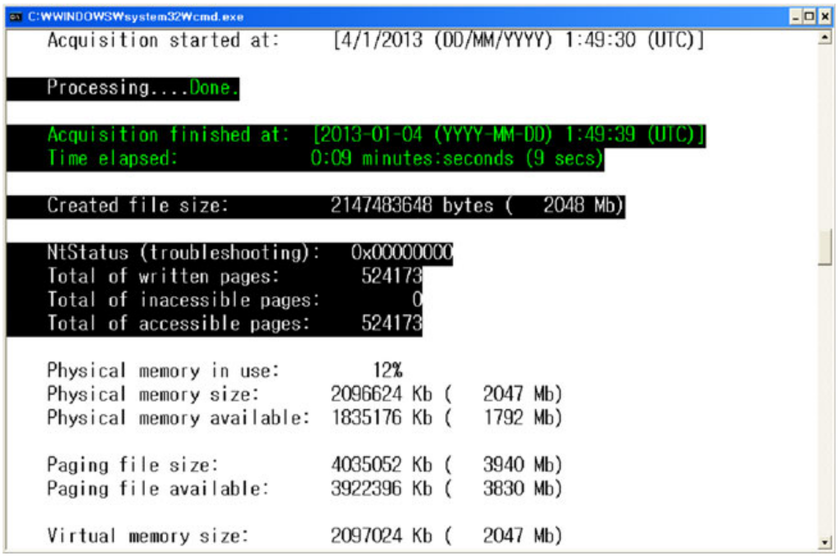
Beginning with Windows Vista, the task manager of the Windows OS provides the process dump of active processes as a default function. It is simple to use. Click the right mouse button on the Windows task bar and select the task manager. Select the Process tab in the Task Manager and click the right mouse button on a process, then select 'Create Dump File' to record the process into a file. The path of the created file is displayed Fig. 2.

2.4.3 Dump using dd (Linux)

dd is one of the default commands of Linux. It provides such functions as copying files of various sizes and changing the format of a specific file. As all devices are managed as files in Linux, the dd command can be used to copy the physical memory data from/dev/mem, which is the memory file. However, in the Linux distribution version limits the access privilege to/dev/mem from the kernel, the access privilege limitation should be released

Table 2 Categorization by memory dump type

	Crash dump tool	Non-crash dump tool - physical memory dump	Non-crash dump - process dump
Target	Whole memory dump Kernel memory dump Mini memory dump	Whole physical memory	Specific processes
Strengths	It is executed during booting so that loss of the memory content is relatively small. As the time of crash dump is prior to the point of using the swap region needed for system booting, it is beneficial when there is a shortage of system resources. It is also used to analyze the cause when a program crashes the system.	Unlike a crash dump tool, it does not require rebooting.	It can dump the memory content of only the target process.
Weakness	Rebooting is needed to enforce memory dump. It is often executed by an unintended error.	Since the process executing the dump is also executed from the memory, obtaining the perfect memory is difficult.	Process execution is terminated in some process dump tools.



```
C:₩WINDOWS₩system32₩cmd.exe
    Acquisition started at:      [4/1/2013 (DD/MM/YYYY) 1:49:30 (UTC)]

    Processing....Done.

    Acquisition finished at:   [2013-01-04 (YYYY-MM-DD) 1:49:39 (UTC)]
    Time elapsed:              0:09 minutes:seconds (9 secs)

    Created file size:          2147483648 bytes (   2048 Mb)

    NtStatus (troubleshooting):    0x00000000
    Total of written pages:          524173
    Total of inacessible pages:           0
    Total of accessible pages:       524173

    Physical memory in use:         12%
    Physical memory size:       2096624 Kb (   2047 Mb)
    Physical memory available:  1835176 Kb (   1792 Mb)

    Paging file size:           4035052 Kb (   3940 Mb)
    Paging file available:      3922396 Kb (   3830 Mb)

    Virtual memory size:        2097024 Kb (   2047 Mb)
```

Fig. 1 Result of memory dump using WinDD

during kernel compiling so that memory dump from the/dev/mem file can be executed.

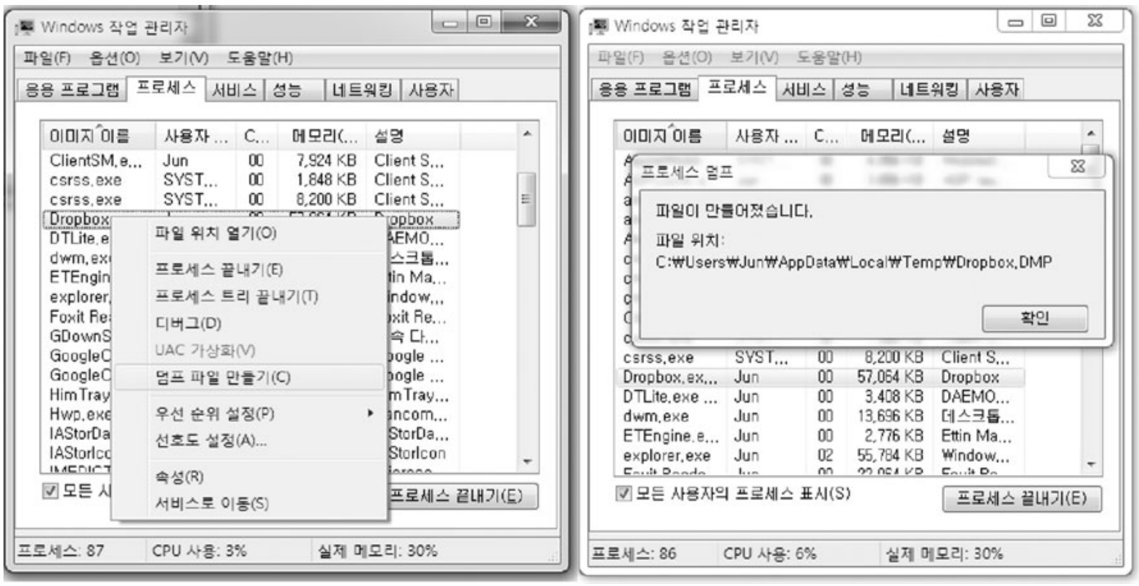
2.4.4 Process dump using gdb (Linux)

gdb is generally included in the Linux distribution version. It is a tool that provides various debugging functions during program development. It can suspend a program at a specific state during execution, and check the status of the variables after an abnormal operation, etc. to help debugging during program development.

Moreover, gdb provides a process dump function based on the memory location of the active process as part of program debugging. To that end, a user first retrieves the list of active processes, checks the pid and memory mapping data of the process, and determines the beginning and ending memory addresses to be dumped. Then, the user runs gdb to dump the memory data of the process.

2.4.5 Lime (Linux)

Lime (Linux Memory Extractor) is a Linux kernel module (KML) first announced at ShmooCon in January 2012 for memory dump of the Linux system. It is the first tool capable of dumping the entire memory in a Linux- or Android-based device. This powerful tool does not require additional operations, such as changing the kernel setting, and can execute memory dump at any time after being compiled. In the case of


Windows 작업 관리자
파일(F) 옵션(O) 보기(V) 도움말(H)
응용 프로그램 | 프로세스 | 서비스 | 성능 | 네트워킹 | 사용자
이미지 이름 | 사용자 ... | C... | 메모리(... | 설명
파일 위치 열기(O)
프로세스 끝내기(E)
프로세스 트리 끝내기(T)
디버그(D)
UAC 가상화(V)
덤프 파일 만들기(C)
우선 순위 설정(P)
선호도 설정(A)...
속성(R)
서비스로 이동(S)
모든 사용자의 프로세스 표시(S)
프로세스 끝내기(E)
프로세스: 87 | CPU 사용: 3% | 실제 메모리: 30%
프로세스 덤프
확인
프로세스: 86 | CPU 사용: 6% | 실제 메모리: 30%

파일이 만들어졌습니다.
파일 위치:
C:₩Users₩Jun₩AppData₩Local₩Temp₩Dropbox.DMP

Fig. 2 Process dump using task manager

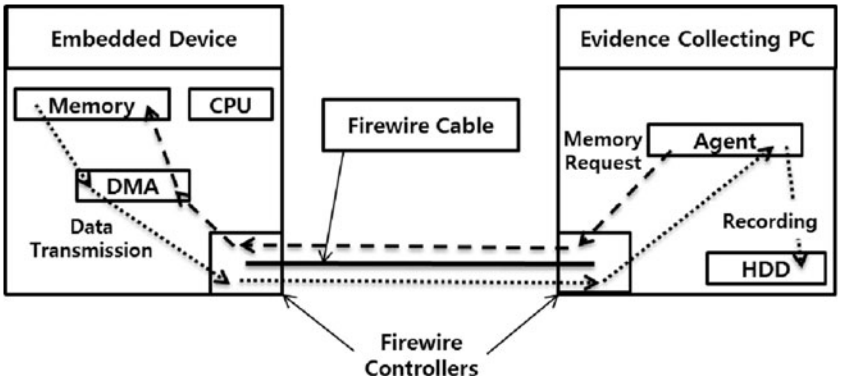


Fig. 3 Mimetic diagram of memory dump using firewire controller

the Android version, if the compiled module file is stored in an external memory, the dumped file can be stored in the external memory.

2.4.6 Dump using hardware

A leading method of memory dump using hardware consists in sending the memory to a DMA controller through the PCI bus in order to acquire a part or all of the memory. Adam Boileau's presentation titled *<Hit By A Bus: Physical Access Attacks with Firewire>* showed that the system memory can be retrieved by sending a DMA request through the Firewire interface. This technique is shown in Fig. 3 below. According to Boileau's paper, memory input/output was normally operated in MAC OS X by Apple; system crash was sometimes caused in some versions of Windows; and writing to memory was possible, while reading from memory was not, in Linux. The problems seemed to be related either to hardware or to the Firewire driver. As such, memory dump through Firewire should be enabled after more studies.

2.4.7 Dump using Vmware virtualization

VMware software enables a user to configure an independent system without the addition of a physical system. Using VMware software, a user can secure the physical memory of the virtually- operated system in one of two ways: One is to dump it with the physical memory software in the system running with VMware; another is to stop the operation of the virtual system and collect the .vmem file, which is the physical memory file. A user can collect .vmem after suspending the operation of the virtual system and either copy the .vmem file or back it up using the system backup function of VMware. During system backup, the snapshot files and the whole memory file (.vmem) are maintained for future restoration, as shown in Fig. 4.

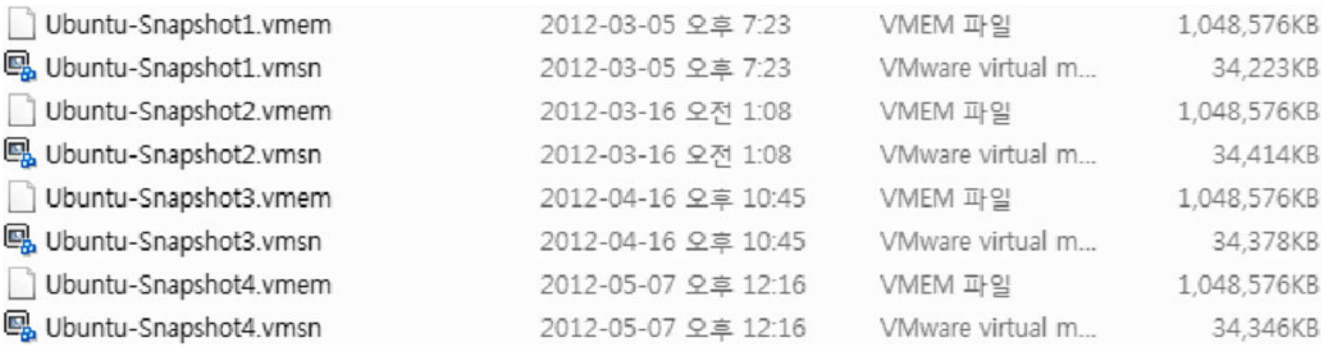
3 Analysis of memory dump in different Oss

3.1 Windows-based memory dump analysis

3.1.1 Analysis of active physical memory using WinHex

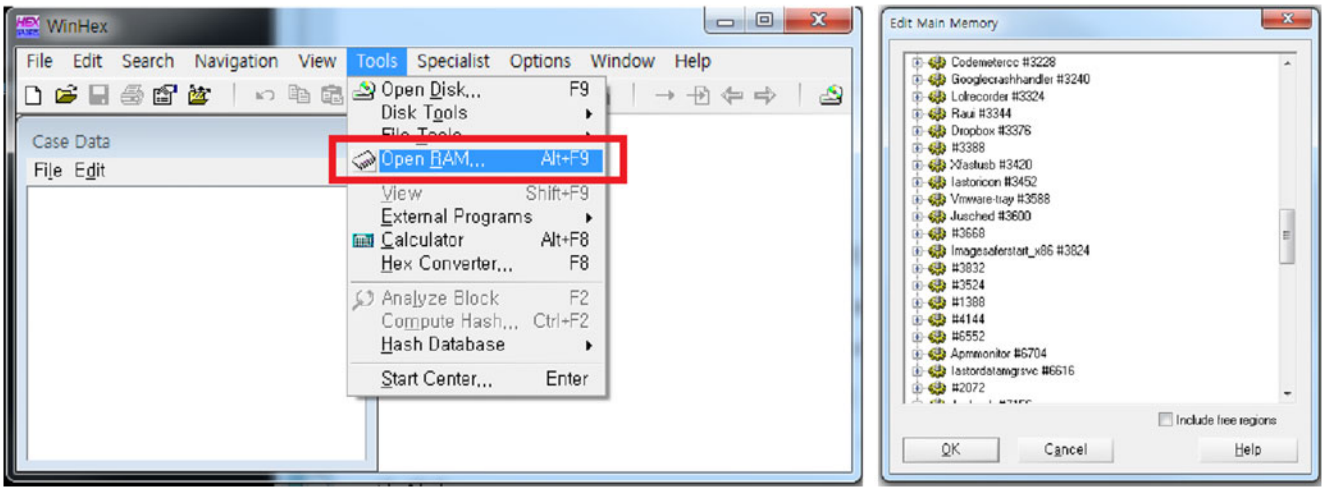
WinHex is a commercial program distributed by X-ways software Technology AG. The hexa editor features various editor functions. One of the tools included with the package is the 'Open RAM' function for checking the physical memory data contained in the hex data in real time. The latest version, WinHex 16.6., runs in Windows OS, but some of its functions are restricted in Windows XP or higher versions Fig. 5.

The strength of Open RAM is that it can monitor the contents of currently active processes or the whole memory in real-time. In particular, it can check the content of the whole memory used by a specific process or the modules included in the process. The Open RAM function for



Ubuntu-Snapshot1.vmem	2012-03-05 오후 7:23	VMEM 파일	1,048,576KB
Ubuntu-Snapshot1.vmsn	2012-03-05 오후 7:23	VMware virtual m...	34,223KB
Ubuntu-Snapshot2.vmem	2012-03-16 오전 1:08	VMEM 파일	1,048,576KB
Ubuntu-Snapshot2.vmsn	2012-03-16 오전 1:08	VMware virtual m...	34,414KB
Ubuntu-Snapshot3.vmem	2012-04-16 오후 10:45	VMEM 파일	1,048,576KB
Ubuntu-Snapshot3.vmsn	2012-04-16 오후 10:45	VMware virtual m...	34,378KB
Ubuntu-Snapshot4.vmem	2012-05-07 오후 12:16	VMEM 파일	1,048,576KB
Ubuntu-Snapshot4.vmsn	2012-05-07 오후 12:16	VMware virtual m...	34,346KB

Fig. 4 VMware snapshot and physical memory file



```
Codemetercc #3228
Googlecrashhandler #3240
Lolrecorder #3324
Raui #3344
Dropbox #3376
#3388
Xfastusb #3420
Iastoricon #3452
Vmware-tray #3588
Jusched #3600
#3668
Imagesaferstart_x86 #3824
#3832
#3524
#1388
#4144
#6552
Apmmonitor #6704
Iastordatamgrsvc #6616
#2072
```

Fig. 5 OpenRAM function of WinHex

checking the whole physical memory is available only in the Windows XP version. When tracing a specific parameter like the password of an account, users can search for a character string in the memory domain of the process.

3.1.2 Volatility framework

Volatility Framework was distributed in open source by Aaron Walters in 2007. Volatility began as the Python-based parsing of memory dump in Windows XP. Then, Volatility 2.0 supported not only Windows XP but also the Windows 2003 Server, Vista, 2008 Server, and Windows 7 32-bit versions. Volatility 2.1 also supported the 64-bit versions of the products supported by version 2.0. The project is still actively ongoing. The latest version, Volatility 2.2 RC2, was issued on September 24, 2012 and became officially available in October 2012. Version 2.2 will also support the Linux 32bit as well as the 64bit kernel 2.6.11 to kernel 3.5 versions. Its roadmap includes version 2.3 in order to support MAC OS, Android/Arm as of the end of 2012.

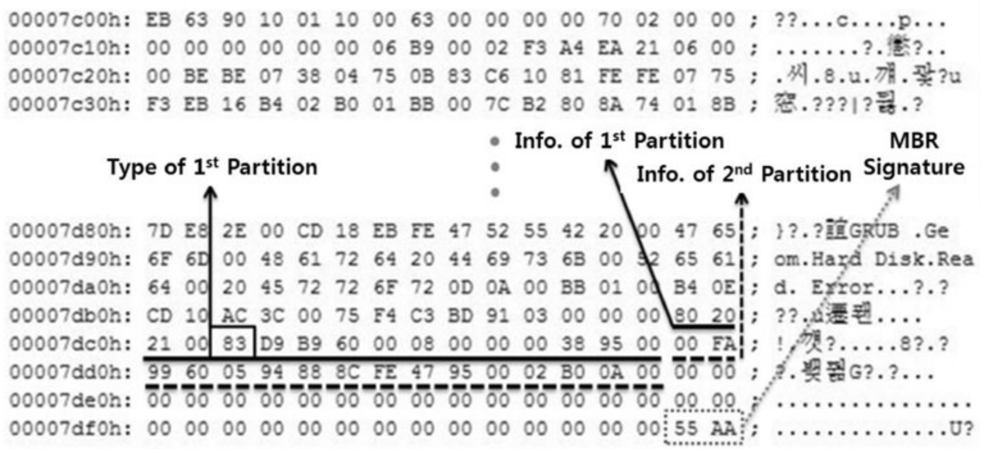
Its main functions include parsing of the memory dump file to extract a list of active processes at the time of dump, identification of loaded modules and drivers as well as files in use, and detection of malicious codes. Moreover, the source code of the program is also available under the GNU GPL v2 license, thus enabling its future expansion.

3.2 Linux-based memory dump analysis

3.2.1 Checking of the process list (Volatilitux)

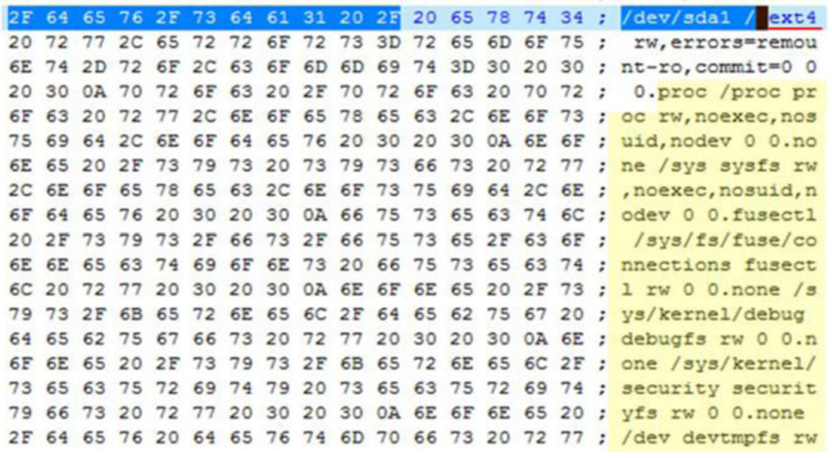
Volatilitux is a GNU GPL v2 license program developed by GIRAULT Emilien, who got the idea from Volatility Framework and created the program to support Linux memory analysis. However, the Volatilitux 1.0 program has not been updated since it was first issued in December 2010. Its supported functions include a list of the active processes at the time of dump, process memory map analysis, process memory dump, open file list extraction, and open file dump. However, as the current Volatility Framework 2.2 version

Fig. 6 MBR partition table in memory dump



```
00007c00h: EB 63 90 10 01 10 00 63 00 00 00 00 70 02 00 00 ; ??...c....p...
00007c10h: 00 00 00 00 00 00 06 B9 00 02 F3 A4 EA 21 06 00 ; .......?.懲?..
00007c20h: 00 BE BE 07 38 04 75 0B 83 C6 10 81 FE FE 07 75 ; .씨.8.u.개.꾳?u
00007c30h: F3 EB 16 B4 02 B0 01 BB 00 7C B2 80 8A 74 01 8B ; 켰.???|?뙋.?
```

```
00007d80h: 7D E8 2E 00 CD 18 EB FE 47 52 55 42 20 00 47 65 ; }?.?頭GRUB .Ge
00007d90h: 6F 6D 00 48 61 72 64 20 44 69 73 6B 00 52 65 61 ; om.Hard Disk.Rea
00007da0h: 64 00 20 45 72 72 6F 72 0D 0A 00 BB 01 00 B4 0E ; d. Error...?.?
00007db0h: CD 10 AC 3C 00 75 F4 C3 BD 91 03 00 00 00 80 20 ; ??.u濩퀀....
00007dc0h: 21 00 83 D9 B9 60 00 08 00 00 00 38 95 00 00 FA ; !.꾳?.....8?.?
00007dd0h: 99 60 05 94 88 8C FE 47 95 00 02 B0 0A 00 00 00 ; ?.뙋뚬G?.?...
00007de0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00007df0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA ; ..............U?
```



```
2F 64 65 76 2F 73 64 61 31 20 2F 20 65 78 74 34 ; /dev/sda1 / ext4
20 72 77 2C 65 72 72 6F 72 73 3D 72 65 6D 6F 75 ;  rw,errors=remou
6E 74 2D 72 6F 2C 63 6F 6D 6D 69 74 3D 30 20 30 ; nt-ro,commit=0 0
20 30 0A 70 72 6F 63 20 2F 70 72 6F 63 20 70 72 ;  0.proc /proc pr
6F 63 20 72 77 2C 6E 6F 65 78 65 63 2C 6E 6F 73 ; oc rw,noexec,nos
75 69 64 2C 6E 6F 64 65 76 20 30 20 30 0A 6E 6F ; uid,nodev 0 0.no
6E 65 20 2F 73 79 73 20 73 79 73 66 73 20 72 77 ; ne /sys sysfs rw
2C 6E 6F 65 78 65 63 2C 6E 6F 73 75 69 64 2C 6E ; ,noexec,nosuid,n
6F 64 65 76 20 30 20 30 0A 66 75 73 65 63 74 6C ; odev 0 0.fusectl
20 2F 73 79 73 2F 66 73 2F 66 75 73 65 2F 63 6F ;  /sys/fs/fuse/co
6E 6E 65 63 74 69 6F 6E 73 20 66 75 73 65 63 74 ; nnections fusect
6C 20 72 77 20 30 20 30 0A 6E 6F 6E 65 20 2F 73 ; l rw 0 0.none /s
79 73 2F 6B 65 72 6E 65 6C 2F 64 65 62 75 67 20 ; ys/kernel/debug 
64 65 62 75 67 66 73 20 72 77 20 30 20 30 0A 6E ; debugfs rw 0 0.n
6F 6E 65 20 2F 73 79 73 2F 6B 65 72 6E 65 6C 2F ; one /sys/kernel/
73 65 63 75 72 69 74 79 20 73 65 63 75 72 69 74 ; security securit
79 66 73 20 72 77 20 30 20 30 0A 6E 6F 6E 65 20 ; yfs rw 0 0.none 
2F 64 65 76 20 64 65 76 74 6D 70 66 73 20 72 77 ; /dev devtmpfs rw
```

Fig. 7 Part of/etc/mtab file found by searching the root file system mount point

supports Linux memory analysis, it is expected that Volatilitux will be used less in the future.

3.2.2 Memory dump analysis for character string search

Since the study of memory dump in Linux is not as active as in Windows, there are very few automation tools. Users can obtain the memory dump file and search the signature character string using a hexa editor to analyze important data. The method can be used to identify the OS and the partition table, to search for the device identification data of the mounted devices, or to extract personal information such as the user's IDs in the Internet records or the Linux user account and password.

4 Testing and results

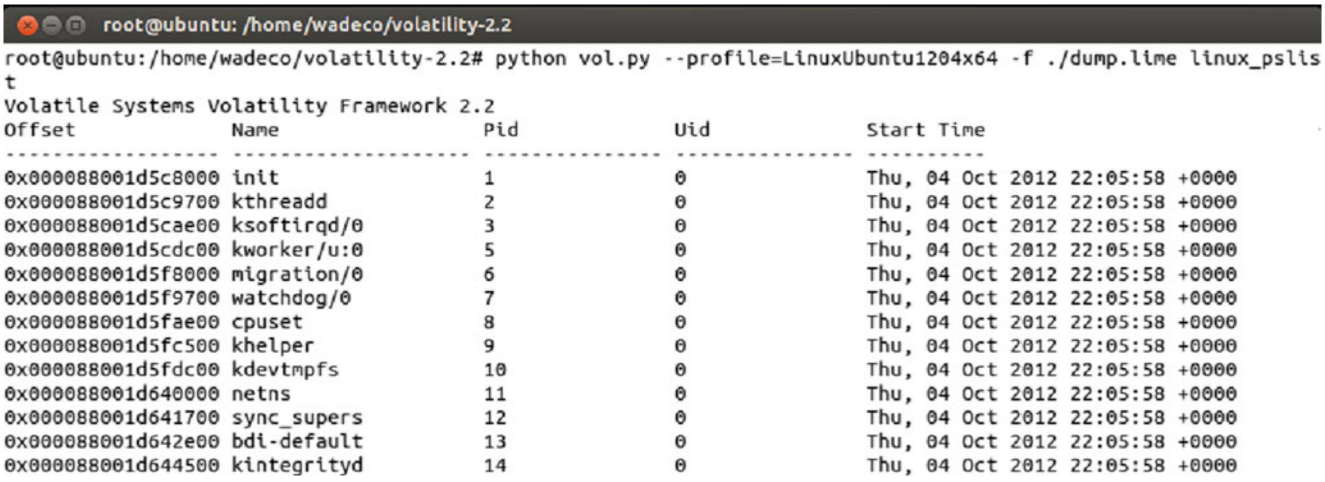
4.1 Test overview

The system used for testing of the memory analysis was a dual core CPU and 256 MB memory with Ubuntu Linux OS version 11.04. Although most of special-purpose system uses a lighter embedded OS, a conventional system was used for the test in order to test subjects applicable not only to the embedded systems but also to conventional Linux systems. The memory was dumped into a raw format using the latest LiME tool. The dumped data were tested using the hexa editor and analysis tool in the desktop of analysis case 3.

4.2 Case study

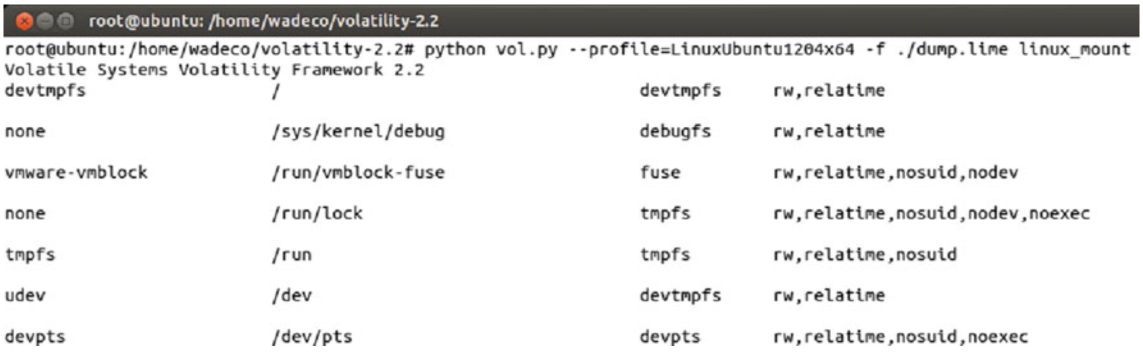
4.2.1 OS and file system checking (case 1)

Identification of the OS is the basic element of digital forensics as it not only checks the OS that is currently running but also allows more information to be gathered from the current OS. Although there are many ways of identifying the OS, the easiest method consists in identifying the partition table of the MBR (Master Boot Record). The MBR is located in the 512byte domain of the first sector of the default storage unit. The domain stores the boot-related code and the partition table. It is loaded into the memory as part of the OS load



```
root@ubuntu: /home/wadeco/volatility-2.2
root@ubuntu:/home/wadeco/volatility-2.2# python vol.py --profile=LinuxUbuntu1204x64 -f ./dump.lime linux_pslis
t
Volatile Systems Volatility Framework 2.2
Offset             Name                 Pid             Uid             Start Time
------------------ -------------------- --------------- --------------- ----------
0x000088001d5c8000 init                 1               0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d5c9700 kthreadd             2               0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d5cae00 ksoftirqd/0          3               0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d5cdc00 kworker/u:0          5               0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d5f8000 migration/0          6               0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d5f9700 watchdog/0           7               0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d5fae00 cpuset               8               0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d5fc500 khelper              9               0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d5fdc00 kdevtmpfs            10              0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d640000 netns                11              0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d641700 sync_supers          12              0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d642e00 bdi-default          13              0               Thu, 04 Oct 2012 22:05:58 +0000
0x000088001d644500 kintegrityd          14              0               Thu, 04 Oct 2012 22:05:58 +0000
```

Fig. 8 Process list retrieved with Volatility 2.2



```
root@ubuntu: /home/wadeco/volatility-2.2
root@ubuntu:/home/wadeco/volatility-2.2# python vol.py --profile=LinuxUbuntu1204x64 -f ./dump.lime linux_mount
Volatile Systems Volatility Framework 2.2
devtmpfs                /                               devtmpfs   rw,relatime

none                    /sys/kernel/debug               debugfs    rw,relatime

vmware-vmblock          /run/vmblock-fuse               fuse       rw,relatime,nosuid,nodev

none                    /run/lock                       tmpfs      rw,relatime,nosuid,nodev,noexec

tmpfs                   /run                            tmpfs      rw,relatime,nosuid

udev                    /dev                            devtmpfs   rw,relatime

devpts                  /dev/pts                        devpts     rw,relatime,nosuid,noexec
```

Fig. 9 Mount data retrieved with Volatility 2.2

during system booting. Therefore, the type of OS of the system can be determined by identifying the boot partition.

The partition table of the MBR is designed to recognize four partitions as the default. Each partition is expressed as 16byte data. The first byte is the 'boot indicator'. If the value is 0×00, it means that the partition cannot be booted. A bootable partition is indicated by the value of 0×80. The fifth byte shows the type of partition, and indicates whether the partition is a DOS FAT, exFAT, Windows NT NTFS, Linux partition or Linux swap partition. One can easily find their type code over the Internet. The following figure shows the partition table contained in the memory dump file of the tested system. It shows the first bootable partition, whose type is 83. Type 83 indicates the "Linux native file system (ex2fs/xiafs)". Thus, checking of the partition type shows that the Linux is the system of the memory dump file. Simply knowing the OS alone can significantly narrow down the number of cases to be tried in an investigation Fig. 6.

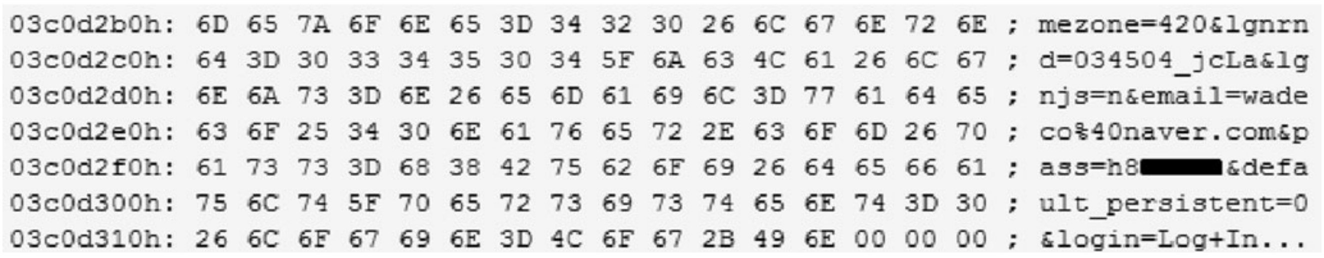
4.2.2 Mount data checking (case 2)

The/etc/mtab file is also loaded into the physical memory when the kernel is loaded into the memory during booting. Therefore, the/etc/mtab file can be found by searching the memory using the character string contained in two files with a hexa editor in a Linux system. This shows the file system mounted in the current system.

To find the/etc/mtab file, a character string contained in the mtab file must be searched. The most probable character string is the name of the device mounted in the root domain. Since/dev/hda1 and/dev/sda1 are very likely to be the device in the root domain, users can search them to find the/etc/mtab file from the dump file. Figure 7 shows the result of the search for the/etc/mtab file contained in the memory dump file.

4.2.3 Memory analysis using tools (case 3)

Information contained in the memory can be obtained quickly and easily using the memory dump file analysis tool. Figure 8 shows a list of processes extracted from a Linux memory dump file using Volatility 2.2. Figure 9 shows a list of devices mounted at the time of memory dump using the same tool. As the figures show, the automation tools help users to obtain the needed information accurately and quickly.



```
03c0d2b0h: 6D 65 7A 6F 6E 65 3D 34 32 30 26 6C 67 6E 72 6E ; mezone=420&lgnrn
03c0d2c0h: 64 3D 30 33 34 35 30 34 5F 6A 63 4C 61 26 6C 67 ; d=034504_jcLa&lg
03c0d2d0h: 6E 6A 73 3D 6E 26 65 6D 61 69 6C 3D 77 61 64 65 ; njs=n&email=wade
03c0d2e0h: 63 6F 25 34 30 6E 61 76 65 72 2E 63 6F 6D 26 70 ; co%40naver.com&p
03c0d2f0h: 61 73 73 3D 68 38 42 75 62 6F 69 26 64 65 66 61 ; ass=h8[redacted]&defa
03c0d300h: 75 6C 74 5F 70 65 72 73 69 73 74 65 6E 74 3D 30 ; ult_persistent=0
03c0d310h: 26 6C 6F 67 69 6E 3D 4C 6F 67 2B 49 6E 00 00 00 ; &login=Log+In...
```

Fig. 10 Account and password in plaintext transmitted as HTTP parameters

4.2.4 Extraction of web service ID and password with character string search (case 4)

The physical memory dump file contains a great deal of parameter information transmitted over the Web. The figure below shows the result of memory dump after connecting to a Facebook Web page. The account data can be obtained by searching character strings like ‘password=’, and ‘isLogin’, which are often transmitted as parameters over the Web. The e-mail address, which was used as the login ID, and the password transmitted in plaintext were exposed when ‘pass=’ was searched Fig. 10.

5 Conclusion and future work

This study analyzed measures and tools for gathering and analyzing the content of the physical memory of Windows, Linux-based system and applied them to actual tested devices for verification. The result indicated that there are technologies that could be used to analyze the physical memory in embedded linux devices. However, no studies have been conducted on the development of measures and analysis tools specifically for embedded devices as yet. Since embedded devices are used for diverse special purposes, they use many different operating systems and interfaces. For that reason it is difficult to apply a specific technology to them, which is probably why there are so few studies involving the analysis of evidence specific to embedded devices.

Regarding analyses of the physical memory, it is very important to decide the time to dump and analyze the memory after an event or an accident, because the content of the memory fills up with irrelevant information after a while due to the nature of the volatile memory. Since embedded devices do not have a particularly large memory, the impact will be even more significant. As regards the embedded devices used by industrial infrastructures, there should be a quick response to the problem to prevent damage nationwide. Therefore, studies on monitoring modules to immediately detect abnormal conditions or cyber-attacks against embedded devices are necessary. If the technology for dumping the memory automatically after detection or upon a command from a remote location and for transferring the data to the network is applied, legally effective evidence of high quality will be obtained more quickly.

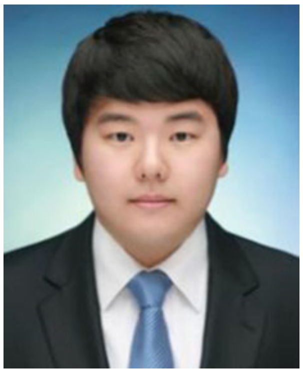
Finally, the procedure for collecting the evidence contained in special-purposed embedded systems as proposed in this paper could be used for the development of digital forensic tools suitable for each evidence collection procedure, which could lead to the standardization of the evidence collection process applied to embedded systems herein.

Acknowledgments This work was supported by the Power Generation & Electricity Delivery of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government Ministry of Trade, industry & Energy (2012101050004A)

References

1. Brendan DG (2008) Forensic analysis of the Windows registry in memory. Digital Investigation, Volume 5, Supplement, S26–S32
2. Vomel S, Freiling FC (2011) A survey of main memory acquisition and analysis techniques for the windows operating system. Digit Investig 8:3–22
3. Petroni NL Jr, Walters AA, Fraser T, Arbaugh WA (2006) FATKit: a framework for the extraction and analysis of digital forensic data from volatile system memory. Digit Investig 3:197–210
4. Han JS, Lee SJ (2011) The windows physical memory dump explorer for live forensics. KIISC J 26(2):71–82
5. Lee SH, Kim HS, Lee SJ, Lim JI (2006) A study of memory information collection and analysis in a view of digital forensic in window system. KIISC J 16(1):87–96
6. Carrier B, Grand J (2004) A hardware-based memory acquisition procedure for digital investigations. Digit Investig 1(1):50–60

Dr. Jung-taek Seo received the Ph.D. degree in information security from Graduate School of Information Management & Security, Korea University, (South) Korea, in 2006. In 2000, he joined the Attached Institute of Electrical and Telecommunication Research Institute, Korea, as a researcher, and he is now a senior researcher at the research institute. His research interests include control system cyber security, cyber security for smart grid, network security, DDoS attack mitigation, etc.



Seokjun Lee received B.E. degree in Information computer engineering from Ajou University in 2011. He is currently studying at Ajou University in Computer Engineering MS and working in information security Lab from 2011. His research interests include Smartgrid Security, Digital Forensic, Mobile Security.

Dr. Taeshik Shon received his Ph.D. degree in Information Security from Korea University, Seoul, Korea and his M.S. and B.S. degree in computer engineering from Ajou University, Suwon, Korea. While he was working toward his Ph.D. degree, he was awarded a KOSEF scholarship to be a research scholar in the Digital Technology Center, University of Minnesota, Minneapolis, USA, from February 2004 to February 2005. From Aug. 2005 to Feb. 2011, Dr. Shon had been a senior engineer in the Convergence S/W Lab, DMC R&D Center of Samsung Electronics Co., Ltd. He is currently a professor at the Division of Information and Computer Engineering, College of Information Technology, Ajou University, Suwon, Korea. He was awarded the Gold Prize for the Sixth Information Security Best Paper Award from the Korea Information Security Agency in 2003, the Honorable Prize for the 24th Student Best Paper Award from Microsoft-KISS, 2005, the Bronze Prize for the Samsung Best Paper Award, 2006, and the Second Level of TRIZ Specialist certification in compliance with the International TRIZ Association requirements, 2008. He is also serving as a guest editor, an editorial staff and review committee of Computers and Electrical Engineering - Elsevier, Mobile Network & Applications - Springer, Security and Communication Networks - Wiley InterScience, Wireless Personal Communications - Springer, Journal of The Korea Institute of Information Security and Cryptology, IAENG International Journal of Computer Science, and other journals. His research interests include Convergence Platform Security, Mobile Cloud Computing Security, Mobile/Wireless Network Security, WPAN/WSN Security, anomaly detection algorithms, and machine learning application.