

Live Memory Forensics of Android Devices

Prepared By

Madhur Tewani

12MCEI29



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INSTITUTE OF TECHNOLOGY, NIRMA UNIVERSITY
AHMEDABAD-382481

May 2014

Live Memory Forensics of Android Devices

Major Project

Submitted in the partial fulfillment of the requirements
for the degree of

Master of Technology in Information and Network Security

Prepared By

Madhur Tewani

12MCEI29

Guided By

Dr. Priyanka Sharma

and

Mr. Nilay Mistry



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INSTITUTE OF TECHNOLOGY, NIRMA UNIVERSITY
AHMEDABAD-382481**

May 2014

Certificate

This is to certify that the major project titled “**Live Memory Forensics of Android Devices**” submitted by Madhur Tewani (12MCEI29), towards the partial fulfillment of the requirements for the degree of Master of Technology in Information and Network Security of Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven’t been submitted to any other university or institution for award of any degree or diploma.

Dr. Priyanka Sharma
Internal Guide & Assoc. Professor,
Dept. of Computer Science & Engg.,
Institute of Technology,
Nirma University,
Ahmedabad

Mr. Nilay Mistry
External Guide & Asst. Professor,
Dept. of Digital Forensics,
Institute of Forensics Science,
Gujarat Forensics Sciences University,
Gandhinagar

Prof. Sharada Valiveti
Assoc. Professor & M.Tech. INS Coordinator,
Dept. of Computer Science & Engg.,
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sanjay Garg
Professor & Head,
Dept. of Computer Science & Engg.,
Institute of Technology,
Nirma University, Ahmedabad

Dr. K. Kotecha,
Director, Institute of Technology, Nirma University, Ahmedabad

Undertaking for Originality of the Work

I, **Madhur Tewani (12MCEI29)**, give undertaking that the Major Project titled **“Live Memory Forensics of Android Devices”** submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Information and Network Security** of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Madhur Tewani (12MCEI29)

Date:

Place:

Endorsed by

Dr. Priyanka Sharma

Internal Guide

Mr. Nilay Mistry

External Guide

Acknowledgments

It gives me immense pleasure in expressing thanks and profound gratitude to **Dr. Priyanka Sharma**, Internal Guide & Associate Professor, Dept. of Computer Science & Engg., Institute of Technology, Nirma University, Ahmedabad and **Mr. Nilay Mistry**, External Guide & Assistant Professor, Dept. of Digital Forensics, Institute of Forensics Science, Gujarat Forensics Sciences University, Gandhinagar for their valuable guidance and continual encouragement throughout this work. The appreciation and continual support they have imparted has been a great motivation to me in reaching a higher goal. Their guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

My deepest thank you is extended to **Prof. Sharada Valiveti**, Coordinator M.Tech. - INS, Dept. of Computer Science & Engg., Institute of Technology, Nirma University, Ahmedabad for an exceptional support and continual encouragement throughout the Major Project.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Dept. of Computer Science & Engg., Institute of Technology, Nirma University, Ahmedabad, **Dr. K. Kotecha**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad and **Dr. M S Dahiya**, Hon'ble Director, Institute of Forensics Science, Gujarat Forensics Sciences University, Gandhinagar for their kind support and providing basic infrastructure and healthy research environment.

I would also thank my colleague friends, the institution and all faculty members of Dept. of Computer Science & Engg., Institute of Technology, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- **Madhur Tewani**

12MCEI29

Abstract

In recent years, with the increase in number of devices running on android operating system, the development of tools for forensics of android devices has been a hot topic of interest among few researchers. This has brought to light, the need of efficient forensics tools specially developed for android devices. This report gives a basic introduction of live forensics of android devices. Then the study about the existing methodologies & tools for forensics of android devices. Finally the methodology of live forensics with practical results of the methods developed for RAM acquisition and its analysis.

Keywords: Android, Live memory forensics, Live memory analysis, Linux, Mobile device live forensics, File carving.

Contents

Certificate	iii
Undertaking	iv
Acknowledgments	v
Abstract	vi
Contents	vi
List of Figures	ix
1 Introduction	1
1.1 Problem Definition	2
1.2 Scope	2
1.3 Background	2
1.4 Chapter Organization	3
2 Literature Survey	4
2.1 Traditional Linux Forensics	4
2.2 Per-process Forensics	4
2.3 Full RAM Forensics	5
3 Implementation Methodology	6
3.1 RAM acquisition process	7
3.2 Analysis of acquired RAM	8
4 Results	10
4.1 Results of unsuccessful methods for RAM acquisition	10
4.1.1 RAM acquisition using dd command	10
4.1.2 RAM acquisition using exploits of UFED	11
4.2 Results of successful method for RAM acquisition	13
4.3 Results of analysis of acquired RAM	14
4.4 Automated script	17

5	Conclusions	19
5.1	Summary	19
5.2	Future Scope	19

List of Figures

3.1	Implementation Methodology	6
3.2	Mount point in case of Aakash Tablet UbiSlate 7Ci	7
3.3	Mount point in case of Sony Xperia L C2104	8
4.1	dd command for mem file on Aakash Tablet UbiSlate 7Ci	10
4.2	RAM acquisition using dd viewed in wxHexEditor	11
4.3	RAM acquisition on non-rooted using UFED files	12
4.4	RAM acquisition on rooted using UFED files	13
4.5	RAM acquisition on rooted Aakash Tablet UbiSlate 7Ci	14
4.6	RAM acquisition on non-rooted Sony Xperia L C2104	14
4.7	Sample configuration file for both foremost and scalpel	15
4.8	Output log file of file carving using foremost	16
4.9	Output log file of file carving using scalpel	16
4.10	Script executed with non-rooted device	18
4.11	Script executed with non-rooted device	18

Chapter 1

Introduction

Forensics means finding the traces from the evidence of how the crime was performed. Digital forensics is the branch of forensics, in which we need to analyze the data of the digital evidences (digital devices). There are two kind of digital forensics viz., live forensics and dead forensics; which can be done either software based or hardware based. Software based digital forensics means the data acquisition, its analysis and all other process is done with the help of software only while hardware based digital forensics means the same procedure done with the inclusion of hardware tools or kits.

Live forensics means analysis of the data is to be done while the device is still powered on, while dead forensics means analysis of the data can be done after the device is powered off. Dead forensics can be done on non-volatile memory like hard-disk drive or any kind of flash drives. RAM is a volatile memory, so it loses the data on restarting the device. Hence live forensics is to be done on RAM instead of dead forensics.

Talking about Android devices, there are two volatile memory viz. RAM and Dalvik Cache that is present in it. Dalvik cache is the cache file generated by the Dalvik Virtual Machine (DVM) stored in RAM itself. DVM is an instance of the Java Virtual Machine (JVM) specially designed for android devices. Dalvik cache is helpful in getting the data particularly for single processes, while RAM has page wise data same as for computers.

1.1 Problem Definition

Developing a methodology for live memory forensics of devices running on Android operating systems. In this process, RAM acquisition of devices running on android operating system is to be done. Acquired data is in hex values which is not readable by humans. After the acquisition of RAM, the acquired data is to be converted to human readable format and then categorized. Here categorizing means to group the data of same type, like all files, all messages, call details, social network application data, etc. The categorized data then can be used to find the traces of crime easily.

1.2 Scope

Each and everyday, number of devices running on android operating system is increasing with a wide variety of devices. Methodologies for dead forensics of android devices are available in a pretty large number. But there are very few methodologies developed for live memory forensics of devices running on Android operating system and also for hand-held devices running on other operating systems, but not so powerful to cover all aspects. This makes a very great scope for developing methodology for live forensics of android devices and other hand-held devices, but android being open-source operating system increases the scope even more for developing live forensics methodology.

1.3 Background

Methodologies currently existing for live forensics of android devices, just can acquire RAM from specific rooted device. Acquisition of RAM is also not possible from non-rooted devices with those methodologies. For analysis of acquired RAM data, hex viewers are to be used as the acquired data will be in hex values. The hex viewers will just show the acquired data in hex values and need to be analyzed manually

only by searching the keywords, app data, sqllitedb files, etc. The other possibility of analyzing data in hex viewer is by carving the files using the header if the hex viewer supports file carving, if not then file carving tools can be used.

1.4 Chapter Organization

This section gives brief information of all the chapters. Chapter 1 gives the introduction about the topic and the report. Chapter 2 contains the literature survey of the existing methodologies and the papers authored by various authors about the live forensics of android devices. Chapter 3 explains the methodology of implementation for whole process of RAM acquisition and analysis of acquired RAM. Chapter 4 shows the results of the method implemented for RAM acquisition and analysis of acquired RAM. Chapter 5 concludes with summary and the future scope.

Chapter 2

Literature Survey

The chapter contains the survey of different tools/methodologies for live forensics of android devices. All these tools/methodologies need the android devices to be rooted for acquisition of data from RAM.

2.1 Traditional Linux Forensics

Traditionally in linux, RAM was stored as a file named **mem** stored at location **/dev/**, but some recent major distributions of linux have this file disabled, same is the case of android. This file used to map only first 896 MB of RAM data. The mem file can be extracted for forensics using various tools, one of which is linux's inbuilt command **dd**. To over come the problem of mem file, **fmem**[1] was developed and it is a loadable kernel module. Full RAM data can be extracted using fmem, but it doesn't work on android devices. Hence, traditional linux forensics methodologies are not successful on android device.

2.2 Per-process Forensics

Some works are done for per-process forensics like **memfetch**[2], **volatilitux**[3] & **memgrab**[4], but each of these have some or the other limitations. Tool 'memgrab'

is helpful for acquisition of data from dalvik cache for specific process only & not full RAM acquisition, so kernel structures, network information and other such data is not included in this memory acquisition. Also one interaction is required for acquisition of each process's data with the device; hence with more no of interaction, increases the possibility of overwriting the user-space data in RAM. Tool 'memfetch' is similar to memgrab which dumps the application data from dalvik cache on demand or on occurrence of fault. Also the unmapped pages won't be dumped by memfetch.

A paper has been authored by Thakur Neha[5] to do forensic analysis of WhatsApp on android devices using memfetch. The result in that paper shows that the forensic analysis was successful upto some extent on rooted device but almost nothing except few encrypted files on non-rooted devices[5]. Deleted data was not recovered using memfetch.

While volatilitux can just provide list of some information like running processes, files currently open, mapping of memory, etc. and cannot provide the full RAM acquisition. So, it can be used along with tools like memfetch or memgrab to list out the process.

2.3 Full RAM Forensics

Moving from per process memory acquisition to whole RAM acquisition, Sylve J, et al.[6], have approached a process for memory acquisition of android devices by loading a module into the kernel and acquire memory. Loading a module in kernel will execute it from kernel space, which doesn't overwrite the user-space data in RAM while execution. The acquired memory can be then analyzed using a tool called volatility, along with a plug-in **LiME** (Linux Memory Extractor)[7] developed by Sylve J.

Chapter 3

Implementation Methodology

The methodology flow for live memory forensics of android device is shown in Fig. 3.1 on page no. 6. The first step is to check whether the android device is rooted or not. Android being linux based operating system, needs root access for RAM acquisition. To check if device is rooted or not, one way is to check if “Superuser” application is installed or not; another way is to check for root shell via Android Debug Bridge (ADB).

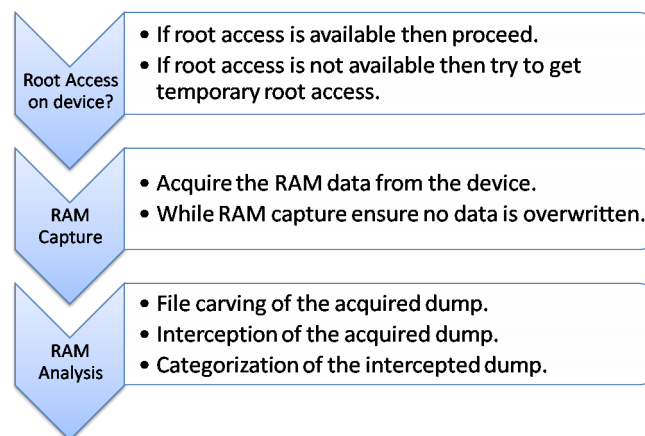


Figure 3.1: Implementation Methodology

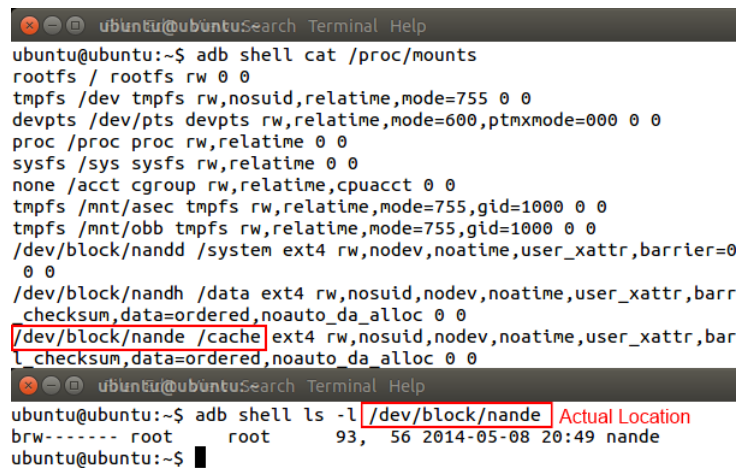
In case the device is not rooted, then the device should not be rooted just before RAM acquisition as device will restart after the rooting process and RAM will be

cleared. So instead of rooting the device, the device needs to be exploited to get the temporary root access. The availability of exploit might not be possible in all cases. After getting the root access on the device, RAM content is to be acquired. Method of RAM acquisition is shown in section 3.1.

The acquired RAM dump will be in hex values and has to be converted to normal ASCII values. After conversion, the dump has to be intercepted like what key terms are found describing about the content. Like for calls there will be key terms about incoming call and outgoing call, similarly for messages and so on. Intercepted output is then to be categorized, like all files of same type together, all the files related to a particular application together and so on.

3.1 RAM acquisition process

In linux, a file called “**mounts**” located in “/proc/” folder, provides a list of all mounts (i.e. file systems mounted) in use by the system with their mount points and other details. This file is present in android devices and also the mount point of the RAM is available in this file.



```

ubuntu@ubuntu:~$ adb shell cat /proc/mounts
rootfs / rootfs rw 0 0
tmpfs /dev tmpfs rw,nosuid,relatime,mode=755 0 0
devpts /dev/pts devpts rw,relatime,mode=600,ptmxmode=000 0 0
proc /proc proc rw,relatime 0 0
sysfs /sys sysfs rw,relatime 0 0
none /acct cgroup rw,relatime,cpuacct 0 0
tmpfs /mnt/asec tmpfs rw,relatime,mode=755,gid=1000 0 0
tmpfs /mnt/obb tmpfs rw,relatime,mode=755,gid=1000 0 0
/dev/block/nandd /system ext4 rw,nodev,noatime,user_xattr,barrier=0
0 0
/dev/block/nandh /data ext4 rw,nosuid,nodev,noatime,user_xattr,barr
checksum,data=ordered,noauto_da_alloc 0 0
/dev/block/nande /cache ext4 rw,nosuid,nodev,noatime,user_xattr,bar
checksum,data=ordered,noauto_da_alloc 0 0
ubuntu@ubuntu:~$ adb shell ls -l /dev/block/nande
brw-rw-rw- root root 93, 56 2014-05-08 20:49 nande

```

Figure 3.2: Mount point in case of Aakash Tablet UbiSlate 7Ci


```

ubuntu@ubuntu:~$ adb shell cat /proc/mounts
rootfs / rootfs ro,relatime 0 0
tmpfs /dev tmpfs rw,nosuid,relatime,mode=755 0 0
devpts /dev/pts devpts rw,relatime,mode=600 0 0
proc /proc proc rw,relatime 0 0
sysfs /sys sysfs rw,relatime 0 0
debugfs /sys/kernel/debug debugfs rw,relatime 0 0
none /acct cgroup rw,relatime,cpuacct 0 0
tmpfs /mnt/secure tmpfs rw,relatime,mode=700 0 0
tmpfs /mnt/asec tmpfs rw,relatime,mode=755,gid=1000 0 0
tmpfs /mnt/obb tmpfs rw,relatime,mode=755,gid=1000 0 0
none /dev/cpuctl cgroup rw,relatime,cpu 0 0
/dev/block/platform/msm_sdcc.1/by-name/system /system ext4 ro,relatime,data=ordered 0
/dev/block/platform/msm_sdcc.1/by-name/userdata /data ext4 rw,nosuid,nodev,relatime,data=ordered 0
/dev/block/platform/msm_sdcc.1/by-name/cache /cache ext4 rw,nosuid,nodev,relatime,data=ordered 0

ubuntu@ubuntu:~$ adb shell ls -l /dev/block/platform/msm_sdcc.1/by-name/cache
lrwxrwxrwx root root 1970-10-12 09:30 cache -> /dev/block/mmcblk0p30
ubuntu@ubuntu:~$ adb shell ls -l /dev/block/mmcblk0p30
brw-rw---- root root 179, 30 1970-10-12 09:30 mmcblk0p30
ubuntu@ubuntu:~$

```

Figure 3.3: Mount point in case of Sony Xperia L C2104

The mount point of RAM in mounts file specifies either the actual location where RAM is mounted or the symbolic link to the location where RAM is mounted depending upon the make of android device. In case of Aakash tablet UbiSlate 7Ci, the mount point is the actual location where RAM is mounted as shown in Fig. 3.2 on page no. 7. While in case of Sony Xperia L C2104, the mount point is of the symbolic link of the actual location where RAM is mounted as shown in Fig. 3.3 on page no. 8.

Making sure that the root access is available in the android device, we can acquire the RAM from the android device by pulling the file at given mount point.

3.2 Analysis of acquired RAM

To analyze the acquired RAM, we need to know what is contained in the RAM. So we need to extract the files from acquired RAM using a file carving technique. Other tools are specific to media files. File carving can be done based on header, structure or contents of the file. File carving based on the header is the most common and

simplest technique used.

For header based file carving, foremost and scalpel are the most famous open-source tools. Foremost was developed by special agents of United States Air Force Office of Special Investigations[8]. Scalpel is originally based on foremost[9]. Both foremost and scalpel, are capable of carving files on providing the header of the required file type in the configuration file. Foremost has built-in function for specific file types. This file carving process will give the files that were present in the RAM of android device at the time of acquisition.

Chapter 4

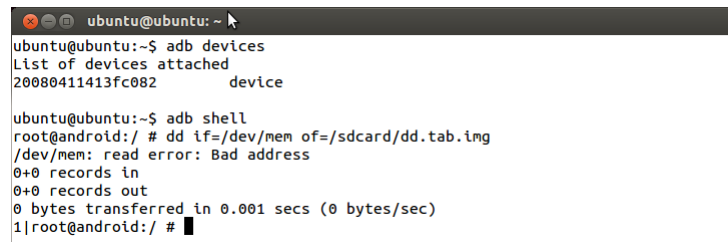
Results

4.1 Results of unsuccessful methods for RAM acquisition

This section contains the results of the methods that were unsuccessful for RAM acquisition on android devices.

4.1.1 RAM acquisition using dd command

RAM acquisition using **dd** command in linux can be done for **mem** file located in **/dev/** folder. The mem file is available in default emulator version of android for Android Virtual Device (AVD).

A terminal window titled 'ubuntu@ubuntu: ~' showing the following commands and output:

```
ubuntu@ubuntu:~$ adb devices
List of devices attached
20080411413fc082    device

ubuntu@ubuntu:~$ adb shell
root@android:/ # dd if=/dev/mem of=/sdcard/dd.tab.img
/dev/mem: read error: Bad address
0+0 records in
0+0 records out
0 bytes transferred in 0.001 secs (0 bytes/sec)
1|root@android:/ #
```

Figure 4.1: dd command for mem file on Aakash Tablet UbiSlate 7Ci

On testing, the RAM acquisition was successful in AVD, but not successful in the case of Aakash Tablet UbiSlate 7Ci, Sony Xperia L C2104 and other android devices. Performing the operation on any of the said devices, gave an error of not able to read the file due to bad address as shown in Fig. 4.1 on page no. 10 for Aakash Tablet UbiSlate 7Ci. Same is the case for other android devices unlike AVD.

The sample output of acquired RAM using dd command from mem file is shown in Fig. 4.2 on page no. 11. The RAM data is in hex values and it is viewed in wxHexEditor open source tool.

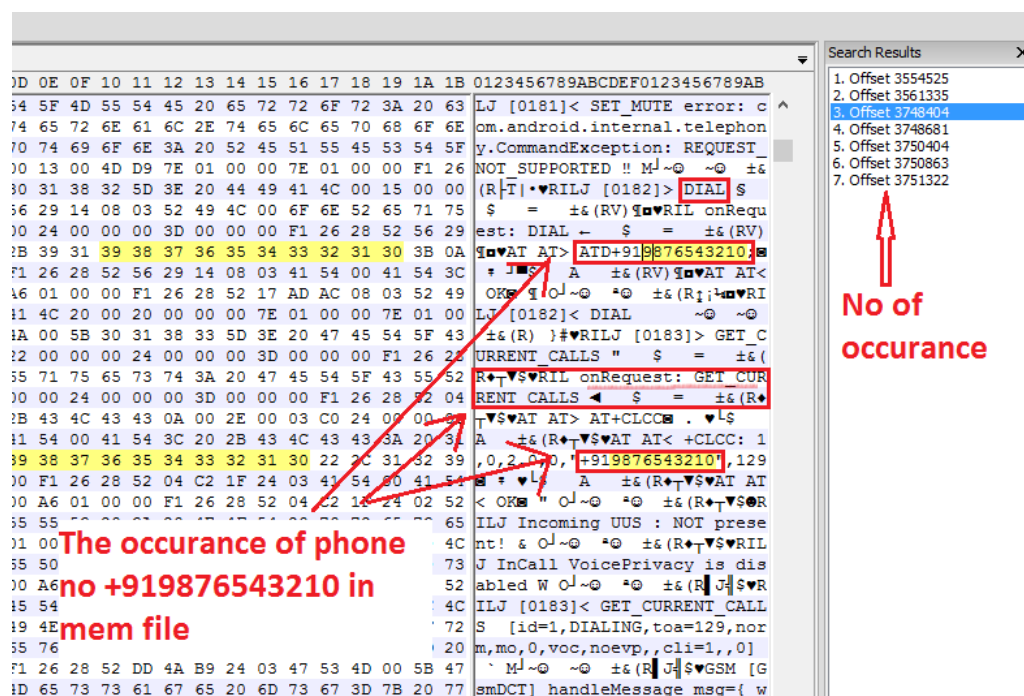
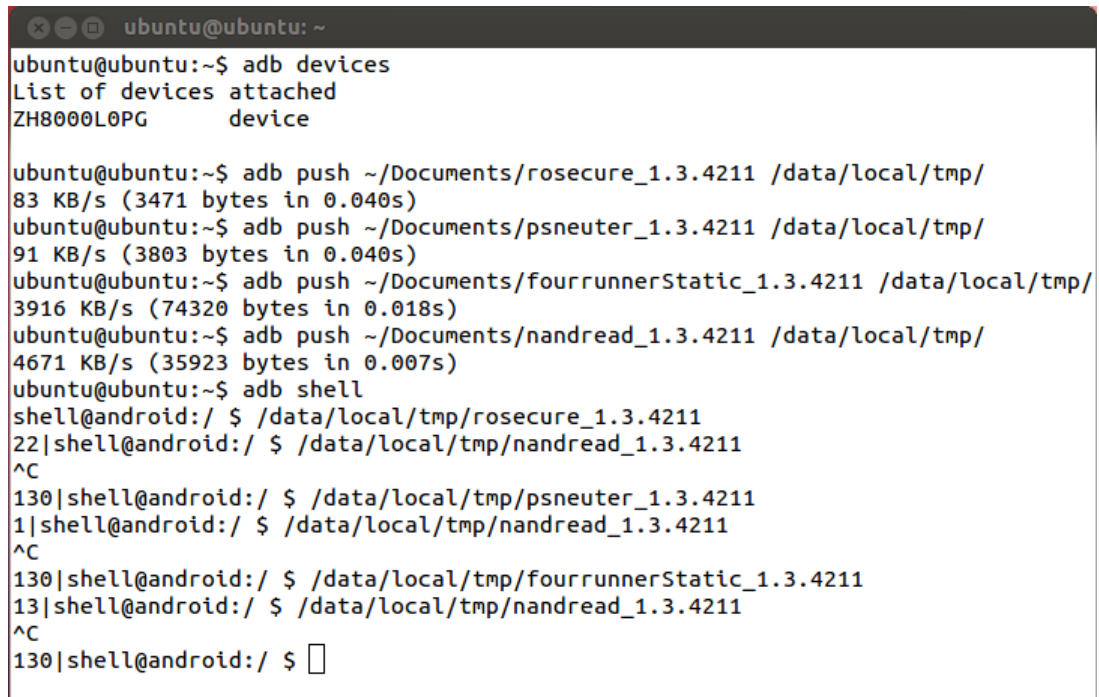


Figure 4.2: RAM acquisition using dd viewed in wxHexEditor

4.1.2 RAM acquisition using exploits of UFED

Universal Forensic Extraction Device (UFED) is an hardware device used for forensics of almost all cell-phones and tablets. UFED has an input port to connect the device of which forensics is to be done and an output port to connect any storage device on



```

ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ adb devices
List of devices attached
ZH8000L0PG      device

ubuntu@ubuntu:~$ adb push ~/Documents/rosecurer_1.3.4211 /data/local/tmp/
83 KB/s (3471 bytes in 0.040s)
ubuntu@ubuntu:~$ adb push ~/Documents/psneuter_1.3.4211 /data/local/tmp/
91 KB/s (3803 bytes in 0.040s)
ubuntu@ubuntu:~$ adb push ~/Documents/fourrunnerStatic_1.3.4211 /data/local/tmp/
3916 KB/s (74320 bytes in 0.018s)
ubuntu@ubuntu:~$ adb push ~/Documents/nandread_1.3.4211 /data/local/tmp/
4671 KB/s (35923 bytes in 0.007s)
ubuntu@ubuntu:~$ adb shell
shell@android:/ $ /data/local/tmp/rosecurer_1.3.4211
22|shell@android:/ $ /data/local/tmp/nandread_1.3.4211
^C
130|shell@android:/ $ /data/local/tmp/psneuter_1.3.4211
1|shell@android:/ $ /data/local/tmp/nandread_1.3.4211
^C
130|shell@android:/ $ /data/local/tmp/fourrunnerStatic_1.3.4211
13|shell@android:/ $ /data/local/tmp/nandread_1.3.4211
^C
130|shell@android:/ $ █

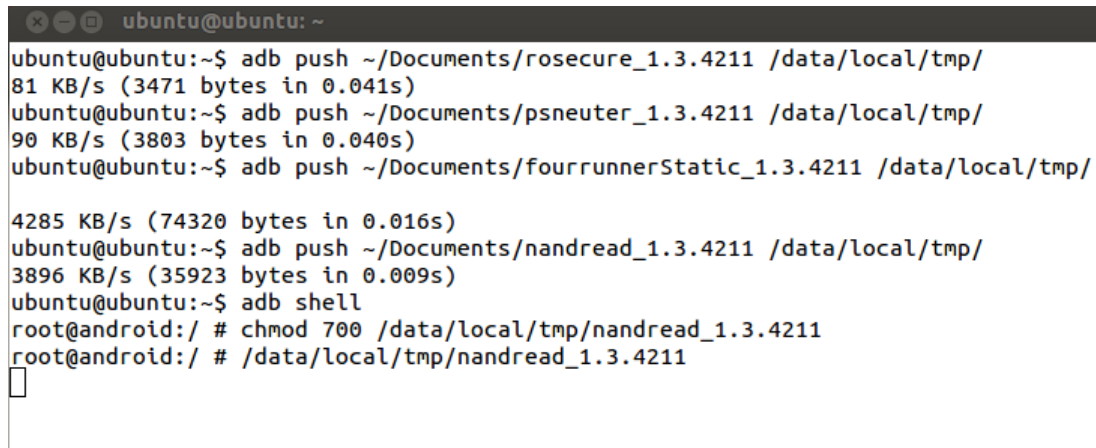
```

Figure 4.3: RAM acquisition on non-rooted using UFED files

which the RAM data of the input device is to be stored.

Studying the log files of UFED it was found that for android devices, UFED pushes some exploits to the android device and executes them from device. These exploits give temporary root access if the device is not rooted. After getting the root access, UFED transfers the RAM data from device to the output storage device. The details of exploits (rosecurer_1.3.4211, psneuter_1.3.4211, fourrunnerStatic_1.3.4211) that are executed by UFED on the input android device are available from the log files for that particular input android device only, also those executable files can be found from the output storage device along with the log file. Other than the details of exploit, the detail for the executable file (nandread_1.3.4211) that transfers RAM data from input android device to output storage device is available from the same log files.

Trying those files manually on non-rooted android device via Android Debug



```

ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ adb push ~/Documents/rosecurer_1.3.4211 /data/local/tmp/
81 KB/s (3471 bytes in 0.041s)
ubuntu@ubuntu:~$ adb push ~/Documents/psneuter_1.3.4211 /data/local/tmp/
90 KB/s (3803 bytes in 0.040s)
ubuntu@ubuntu:~$ adb push ~/Documents/fourrunnerStatic_1.3.4211 /data/local/tmp/
4285 KB/s (74320 bytes in 0.016s)
ubuntu@ubuntu:~$ adb push ~/Documents/nandread_1.3.4211 /data/local/tmp/
3896 KB/s (35923 bytes in 0.009s)
ubuntu@ubuntu:~$ adb shell
root@android:/ # chmod 700 /data/local/tmp/nandread_1.3.4211
root@android:/ # /data/local/tmp/nandread_1.3.4211

```

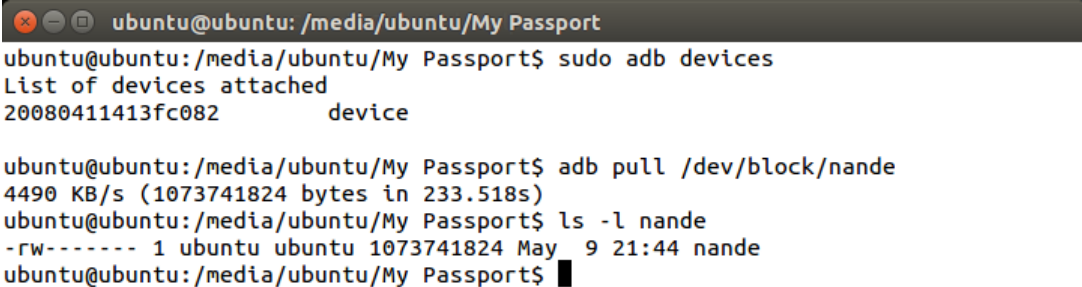
Figure 4.4: RAM acquisition on rooted using UFED files

Bridge (ADB), the exploits doesn't give temporary root access to nandread_1.3.4211 file to transfer RAM data from android device to any output location as shown in Fig. 4.3 on page no. 12. Also the root access after executing the said file is not available on the ADB shell. Trying the nandread_1.3.4211 file manually on already rooted android device doesn't successfully transfer RAM data from android device to any output location as shown in Fig. 4.4 on page no. 13. The reason probably is the file might be hard coded to work on UFED device exclusively.

4.2 Results of successful method for RAM acquisition

This section contains the results of final successful method of RAM acquisition as described and shown in section 3.1 on page no. 7, by searching for mount point in the "/proc/mounts" file of the device. As explained earlier, this step can be performed only on the rooted devices. The sample of the pulling the RAM file from the given mount point is shown in Fig. 4.5 on page no. 14 and Fig. 4.6 on page no. 14 for devices rooted Aakash Tablet UbiSlate 7Ci and non-rooted Sony Xperia L C2104

respectively.



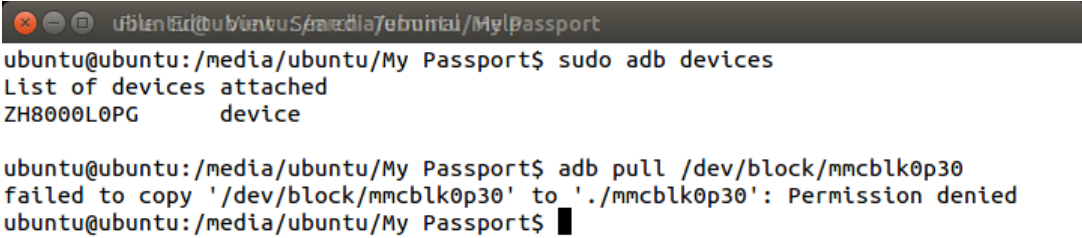
```

ubuntu@ubuntu: /media/ubuntu/My Passport
ubuntu@ubuntu:/media/ubuntu/My Passport$ sudo adb devices
List of devices attached
20080411413fc082      device

ubuntu@ubuntu:/media/ubuntu/My Passport$ adb pull /dev/block/nande
4490 KB/s (1073741824 bytes in 233.518s)
ubuntu@ubuntu:/media/ubuntu/My Passport$ ls -l nande
-rw----- 1 ubuntu ubuntu 1073741824 May  9 21:44 nande
ubuntu@ubuntu:/media/ubuntu/My Passport$

```

Figure 4.5: RAM acquisition on rooted Aakash Tablet UbiSlate 7Ci



```

ubuntu@ubuntu: /media/ubuntu/My Passport
ubuntu@ubuntu:/media/ubuntu/My Passport$ sudo adb devices
List of devices attached
ZH8000L0PG      device

ubuntu@ubuntu:/media/ubuntu/My Passport$ adb pull /dev/block/mmcblk0p30
failed to copy '/dev/block/mmcblk0p30' to './mmcblk0p30': Permission denied
ubuntu@ubuntu:/media/ubuntu/My Passport$

```

Figure 4.6: RAM acquisition on non-rooted Sony Xperia L C2104

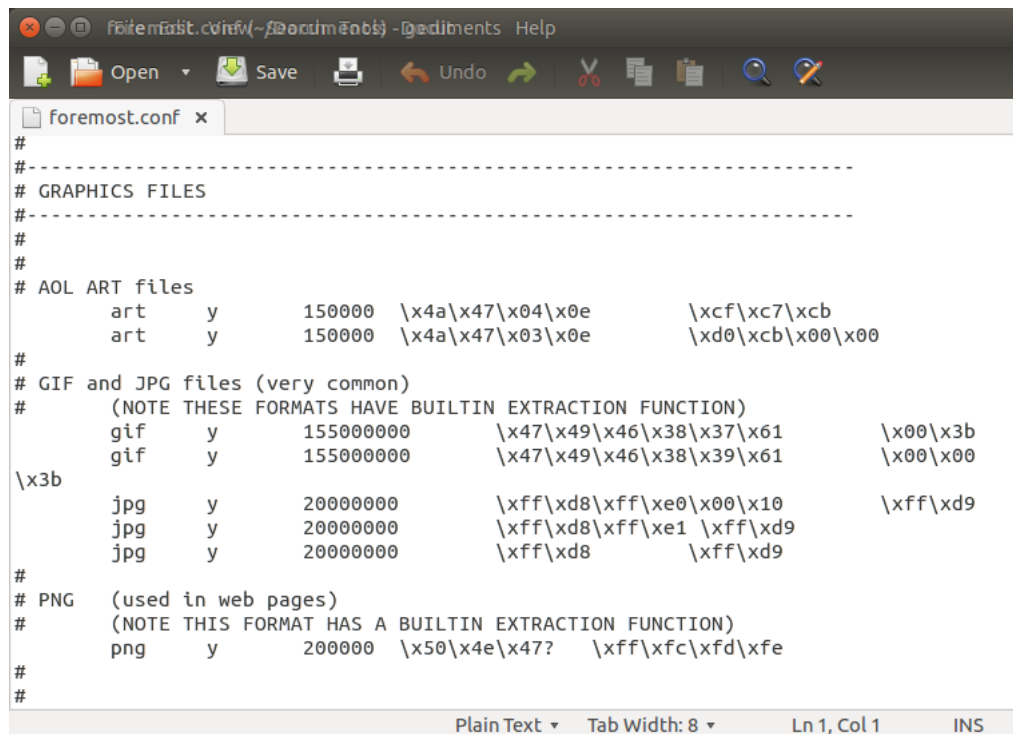
In the case of rooted Aakash Tablet UbiSlate 7Ci, the RAM was acquired while in the case of non-rooted Sony Xperia L C2104, the RAM wasn't acquired and it gave an error of "Permission Denied".

4.3 Results of analysis of acquired RAM

After the acquisition of RAM, the files contained in it need to be extracted for analysis. As explained earlier in section 3.2 on page no. 8, either of the two famous open-source file carving tools (viz. foremost or scalpel) can be used for extraction of files from acquired RAM.

Both foremost and scalpel, have capability of carving files on providing the header of the required file type in the configuration file. Same configuration file can be used

for both. Foremost has built-in function for specific file types. Sample configuration file is shown in Fig. 4.7 on page no. 15.



```
#
#
#-----
# GRAPHICS FILES
#-----
#
#
# AOL ART files
#   art      y      150000  \x4a\x47\x04\x0e      \xcf\xcb\xcb
#   art      y      150000  \x4a\x47\x03\x0e      \xd0\xcb\x00\x00
#
# GIF and JPG files (very common)
# (NOTE THESE FORMATS HAVE BUILTIN EXTRACTION FUNCTION)
#   gif      y      155000000  \x47\x49\x46\x38\x37\x61      \x00\x3b
#   gif      y      155000000  \x47\x49\x46\x38\x39\x61      \x00\x00
#
#   jpg      y      200000000  \xff\xd8\xff\xe0\x00\x10      \xff\xd9
#   jpg      y      200000000  \xff\xd8\xff\xe1 \xff\xd9
#   jpg      y      200000000  \xff\xd8      \xff\xd9
#
# PNG (used in web pages)
# (NOTE THIS FORMAT HAS A BUILTIN EXTRACTION FUNCTION)
#   png      y      200000  \x50\x4e\x47?  \xff\xfc\xfd\xfe
#
#
```

Figure 4.7: Sample configuration file for both foremost and scalpel

For the file types of which foremost has built-in function, scalpel gives a very large number of false positive files and less number of true positive files compared to foremost. While for rest of the file types, the number of false positive and true positive files is almost same for both. Output log file of file carving using foremost and scalpel is shown in Fig. 4.8 on page no. 16 and Fig. 4.9 on page no. 16.

Output log file of foremost shows total 70075 files (9299+60776), while output log file of scalpel shows total 72212 files (72225-13). Both the tools give enormous number of false positive files but scalpel gives more than foremost. Overall, foremost is more feasible compared to scalpel, so foremost is preferred over scalpel and is used further.

The reason for this enormous false positive files is due to fragmentation of the


```
audit.txt x (13 x 32)
9294: 01320615.jpg 3 MB 676155228
9295: 01334224.htm 315 B 683123113
9296: 01334232.htm 67 B 683126898
9297: 01357733.htm 598 B 695159312
9298: 01454254.jpg 304 KB 744578512
Finish: Fri May 9 20:57:34 2014
9299 FILES EXTRACTED
jpg:= 166
gif:= 41
bmp:= 4
rif:= 7
htm:= 24
zip:= 17
png:= 7870
pdf:= 9
db:= 1161
Foremost finished at Fri May 9 20:57:34 2014

audit.txt x
Finish: Fri May 9 21:22:59 2014
60776 FILES EXTRACTED
art:= 1
tif:= 31
mpg:= 28
mpg:= 78
mpg:= 83
fws:= 25
wpc:= 8
pgp:= 15331
pgp:= 8717
pgp:= 13659
pgp:= 13309
rpm:= 1998
ra:= 1
asf:= 1
wmv:= 1
wma:= 1
wma:= 1
dat:= 5
cookie:= 6337
db:= 1161
Foremost finished at Fri May 9 21:23:00 2014
```

Figure 4.8: Output log file of file carving using foremost

```
audit.txt x
12 The following files were carved:
13 File Start Chop Length Extracted From
14 00008285.pgp 2415495 YES 100000 nande
15 00008286.pgp 3064836 YES 100000 nande
16 00008287.pgp 4296715 YES 100000 nande
17 00008288.pgp 4629028 YES 100000 nande
18 00008289.pgp 4976699 YES 100000 nande
19 00023616.pgp 4218891 YES 100000 nande
. . . . .
. . . . .
. . . . .
72220 00007280.mov 747184181 YES 10000000 nande
72221 00007279.mov 747184172 YES 10000000 nande
72222 00007278.mov 747184166 YES 10000000 nande
72223 00007277.mov 747143836 YES 10000000 nande
72224 00007276.mov 746812094 YES 10000000 nande
72225 00007275.mov 746756954 YES 10000000 nande
72226
72227
72228 Completed at Sun Mar 23 16:04:51 2014
```

Figure 4.9: Output log file of file carving using scalpel

files. Due to the file fragmentation, the structure of files also might get destroyed. Like in the case of sqllitedb database files, the content will be there in the file but might not be viewable as a database in sqlite viewer. So to view those files, any hex viewer can be used.

SmartCarving[10] technique to overcome problem of fragmentation has been designed but the tools using this technique have been developed for media files (images and videos) only.

4.4 Automated script

Whole process of live memory forensics as described above is automated using python script. The prerequisites of the automated script is linux machine with following installations:

- a. Android Debug Bridge (ADB) [Either using platform-tools from android sdk[11] or directly from default resources of linux distribution in use.]
- b. PyAdb API[12]
- c. Foremost[13]

After the required installations have been done, the android device is to be connected to the machine with USB debugging mode enabled and the script is to be executed. Fig. 4.10 on page no. 18 and Fig. 4.11 on page no. 18 shows the sample output of the script executed with non-rooted device and rooted device respectively when connected to machine. The non-rooted device used for this was Sony Xperia L C2104 and the rooted device used was Aakash Tablet UbiSlate 7Ci.

```

ubuntu@ubuntu:/media/ubuntu/My Passport$ python adb_version_data.py
[+] PyAdb API Version: 0.1.1
[+] Verifying ADB path... OK
[+] ADB Version: 1.0.31

[+] Restarting ADB server...
[+] Detecting devices... OK
    0: ZH8000L0PG

[+] Using "ZH8000L0PG" as target device
[+] Local directory path for target device : ZH8000L0PG_10052014_053823PM

[+] Capturing RAM file from remote device "ZH8000L0PG"...

[!] Error: failed to copy '/dev/block/mmcblk0p31' to 'ZH8000L0PG_10052014_053823PM/mmcblk0p31': Permission denied

ubuntu@ubuntu:/media/ubuntu/My Passport$ █

```

Figure 4.10: Script executed with non-rooted device

```

ubuntu@ubuntu: /media/ubuntu/My Passport$ sudo python adb_version_data.py
[+] PyAdb API Version: 0.1.1
[+] Verifying ADB path... OK
[+] ADB Version: 1.0.31

[+] Restarting ADB server...
[+] Detecting devices... OK
    0: 20080411413fc082

[+] Using "20080411413fc082" as target device
[+] Local directory path for target device : 20080411413fc082_11052014_024949AM

[+] Capturing RAM file from remote device "20080411413fc082"...
[+] Captured RAM file from remote device "20080411413fc082" and stored in directory
"20080411413fc082_11052014_024949AM"

[+] Carving files from captured RAM file...
[+] Phase 1:
[+] Phase 1 file carving complete.
[+] Phase 1 output files stored in "20080411413fc082_11052014_024949AM/output-p1".
[+] Phase 2:
[+] Phase 2 file carving complete.
[+] Phase 2 output files stored in "20080411413fc082_11052014_024949AM/output-p2".
ubuntu@ubuntu:/media/ubuntu/My Passport$ █

```

Figure 4.11: Script executed with non-rooted device

Chapter 5

Conclusions

5.1 Summary

The methodology of full RAM acquisition has been developed for rooted android devices. The developed methodology doesn't work for non-rooted android devices due to insufficient permissions to access RAM. The extraction of files from acquired RAM has also been done for analysis. Whole process of live memory forensics of android devices, i.e. RAM acquisition and extraction of files from it, has been scripted in python programming language.

5.2 Future Scope

The future work to be done is to implement SmartCarving[10] technique for extraction of files from acquired RAM, after the technique has been properly developed.

Bibliography

- [1] I. Kollár and I. Kollár, “Forensic ram dump image analyser.” Masters, Department of Software Engineering, Charles University, 2010.
- [2] Z. Michal. <http://lcamtuf.coredump.cx/soft/memfetch.tgz>, 2002.
- [3] E. Girault, “Volatilitux : Physical memory analysis of Linux systems.” <http://www.segmentationfault.fr/projets/volatilitux-physical-memory-analysis-linux-systems/>, 2010.
- [4] V. L. L. Thing, K.-Y. Ng, and E.-C. Chang, “Live memory forensics of mobile phones,” *Digit. Investig.*, vol. 7, pp. S74–S82, Aug. 2010.
- [5] N. S. Thakur, “Forensic analysis of whatsapp on android smartphones.” University of New Orleans Theses and Dissertations, 2013.
- [6] J. Sylve, A. Case, L. Marziale, and G. G. Richard, “Acquisition and analysis of volatile memory from android devices,” *Digital Investigation*, vol. 8, no. 34, pp. 175 – 184, 2012.
- [7] J. Sylve, “LiME (Linux Memory Extractor) v1.1 Documentation,” 2013.
- [8] [http://en.wikipedia.org/wiki/Foremost_\(software\)](http://en.wikipedia.org/wiki/Foremost_(software)).
- [9] <http://www.forensicswiki.org/wiki/Scalpel>.
- [10] http://www.forensicswiki.org/wiki/File_Carving:SmartCarving.
- [11] <http://developer.android.com/sdk/index.html>.
- [12] <https://pypi.python.org/pypi/pyadb>.
- [13] <http://foremost.sourceforge.net/>.