International Journal of Cyber-Security and Digital Forensics (IJCSDF) 3(3): 130-140
The Society of Digital Information and Wireless Communications, 2014 (ISSN: 2305-0012)

Performance Measurement for Mobile Forensic Data Acquisition in Firefox OS

Mohd Najwadi Yusoff, Ramlan Mahmod, Mohd Taufik Abdullah, Ali Dehghantanha
Faculty of Computer Science & Information Technology,
Universiti Putra Malaysia,
Serdang, Selangor, Malaysia.
najwadi@cs.usm.my,{ramlan,taufik,alid}@upm.edu.my

ABSTRACT

Mozilla Corporation has recently released a Linux-based open source operating system, namely Firefox OS. The arrival of this Firefox OS has created new challenges, concentrations and opportunities for digital investigators. Currently, Firefox OS is still not fully supported by most of the existing mobile forensic tools. Even when the phone is detected as Android, only pictures from removable memory was able to be captured. Furthermore, the internal data acquisition is still not working. Therefore, there are very huge opportunities to explore the Firefox OS on every stages of mobile forensic procedures. This paper will present an approach for mobile forensic data acquisition in a forensically sound manner from a Firefox OS running device. This approach will largely use the UNIX dd command to create a forensic image from the Firefox OS running device. Apart from that, performance measurement will be made to find the best block size for acquisition process in Firefox OS.

KEYWORDS

Mobile forensic, data acquisition, forensic image, dd command, Firefox OS.

1 INTRODUCTION

The advancement of smartphone technology has attracted many companies in developing their own mobile operating system. Recently released Firefox OS is an open source mobile operating system which is purely based on Linux and Mozilla's Gecko technology [1]. Firefox OS boots into a Gecko-based runtime engine and thus allow users to run applications developed exclusively using *HTML5*, *JavaScript*, and other open web application *APIs*. According to Mozilla Developer Network, Firefox OS is free from proprietary technology, but still a powerful platform; it offers application developers an opportunity to create tremendous products [1]. Mozilla introduced *WebAPI* by bridging the capability gap between native frameworks and web applications. *WebAPI* enable developers to build applications, and run it in any standards compliant browser without the need to rewrite their application for each platform. In addition, since the software stack is entirely *HTML5*, a large number of developers were already established, and users can embrace the freedom of pure *HTML5* [2].

To our knowledge, none of the existing mobile forensic tools are working perfectly with Firefox OS. For example, *MobilEdit!* is able to detect Firefox OS running phone, but listing it as an Android device. When we tried to perform data acquisition, *MobilEdit!* was only capable to acquiring some of the pictures from removable memory, the remaining are left undetected. In addition, we have also tried using *Paraben Device Seizure*, *Oxygen Forensic Suite*, *Cellebrite Mobile Forensics* as well as *Micro Systemation XRY*; and the result were even worse. The acquisition process for Firefox OS running phone become more exciting because the phone itself was detected as Android. This may be due to the similarity of both Android and Firefox OS in their based kernel. For that reason alone, this paper will demonstrate the use of *Android Debug Bridge (ADB)*; to connect the phone with the host machine and acquiring the phone image using UNIX dd command.

There are three types of acquisition of mobile devices; manual, logical and physical [3]. Manual acquisition is defined as the capability of acquiring data by interacting with the device itself.

Logical acquisition is recovering a bitwise copy of entities that reside in a logical storage, and lastly; the physical acquisition is solely related to the physical storage medium. In most cases, manual acquisition takes place simultaneously with the other two acquisition methods. On the contrary, there are strengths and weaknesses of each types of acquisition. Grispos stated that, logical acquisition is more efficient for recovering user data, whereas physical acquisition can retrieve deleted files [4]; but this procedure can damage the device while it is being dismantled. And for that reason, this paper will only perform the combination between manual and logical types of acquisition. During this process, we will be making a bitwise copy of all partitions, and keeping the log of actions taken. After that, we will run a few acquisition test with different block size to find the best block size for acquisition process in Firefox OS. Furthermore, the result will be presented in the graph form and we will recommend the best block size to be used for future investigation in Firefox OS.

The objective of this paper is to present the detail steps on how we manage to acquire mobile forensic image from Firefox OS using UNIX dd command without making any changes to the phone during acquisition. This paper also present performance measurement for block size during acquisition process. This paper is organized as follows; Section (2) will explain about the state of the arts. Section (3) will present acquisition methodology and the detail steps. Section (4) is about performance measurement and Section (5) will give a brief conclusion and the future work to be considered. Acknowledgement and references are also presented at the end of this paper.

2 STATE OF THE ART

2.1 Early Mobile Investigation

Data acquisition is the procedure of imaging and obtaining evidence from a mobile device and its peripheral equipment [5]. In the earliest mobile forensic investigation, most of the digital evidences in mobile phone were stored in SIM cards. Research by Goode stated that, it is vital to acquire the data such as contacts and SMSs stored in SIM cards [6]. In addition, mobile phone memory and SIM cards also hold phone contacts which may contain critical evidences for an investigators. According to Goode, there are three evidence locations in mobile phone which are from SIM cards, identification information from a mobile phone (*IMEI*) and core network provider information. Similar work carried out by Willassen was by exploring SIM card and core network data in GSM phones [7]. According to Willassen, the SIM cards can provide information of network provider name with a unique identification number. The subscriber's name, phone number and address usually associated with the SIM cards. Consequently, phone records can also be retrieved from network providers. Furthermore, the contents of a SIM cards are binary data that can be taken, provided that the user has authentication either with a PIN or a PUK code. Programs or tools such as *Cards4Labs* and *SIM-Surf Profi* were used to decode the binary format into readable form. In addition, Willasen also able to recover the evidence such as phone logs, phone contacts, SMS, and phone *IMEI* obtained from both SIM cards and mobile phones.

In similar attempt, Casadei was used open source tools, both in Windows and Linux for digital extraction from SIM cards [8]. As the result, Casadei was able to acquire the raw data in Binary format from the SIM cards. Casadei also presented an interpretation of binary raw data at a higher level of abstraction and used an open source tool named *SIMbrush* to examine the raw data. *SIMbrush* was designed to acquire digital evidence from any SIM cards in GSM network but have not tested for any modules under *D-AMPS*, *CDMA* and *PDC*. Additionally, *SIMbrush* focus more towards GSM network because GSM is the biggest mobile network in the world at that time and penetration in this network is rapidly increased. Marturana was extended the acquisition process in SIM cards by comparing data in SIM cards and smartphones [9]. According to Marturana, acquisition in the smartphone is much more complicated; this is due to the possibility of evidences are also stored in many places such as internal and flash memory.

2.2 Revolutionary of Smartphone

With the emergence of smartphones, focuses are more on the Windows Mobile OS due to its similarity in nature with the desktop environment. Windows Mobile OS is a simplified version of Windows OS developed by Microsoft; mainly for mobile devices. Research by Chen was able to extract SMS, phone book, call recording, scheduling, and documents from Windows Mobile OS via Bluetooth, Infrared and USB mode using *Microsoft ActiveSync* [10]. *Microsoft ActiveSync* used *Remote API* (*RAPI*) to manage, control, and interact with the connection equipment from the desktop computer. The acquired data were came from mobile phone internal memory, SIM card as well as removable memory. Similar research was continued by Irwin and Hunt by extracting evidences over wireless connections. They used their own developed forensic tools called as *DataGrabber*, *CTASms* and *SDCap*. *DataGrabber* was used to retrieve information from both the internal memory and any external storage card, *CTASms* to extract information from the mobile device's Personal Information Manager (*PIM*) while *SDCap* was used to extract all information from external storage card. They were successfully mapping internal and external phone's memory and transfer all files and folder to desktop computers [11].

By using *RAPI* function, acquisition process only capable to capture active data and capturing deleted data is not possible using this method. According to Klaver, physical acquisition method will be able to obtain non-active data in Windows Mobile OS [12]. Klaver was proposed a versatile method to investigate isolated volume of Windows Mobile OS database files for both active and deleted data. Klaver was used freely available tools of forensic application and explained the known methods of physical acquisition. Deleted data can be recovered by using advanced acquisition methods like chip extraction and this method was able to bypass password protection. Casey was extended the finding by describing various methods of acquiring and examining data on Windows Mobile devices. Casey was also able to capture text messages, multimedia, e-mail, Web browsing, and Registry entries [13]. Some of the captured data by Casey were locked by the OS itself, and require *XACT* from *Micro Systemation* and *ItsUtils* to work together with *Microsoft ActiveSync*. These tools will help to unlock certain files and convert the *ASCII* format in *cemail.vol* structure to a readable SMS. This research was also focused on potentially useful sources of evidences in Windows Mobile OS and addressed the potential evidences found in *"\temp"* folder. In the recent work, Kaart made an investigation by reverse-engineering the *pim.vol* volume files in Windows Mobile OS [12]. *pim.vol* is a Microsoft's Embedded Database (*EDB*) volume that consists of information related to phone contacts, calendars, appointments, call history, speed-dial settings and tasks [12]. Kaart was successfully reverse-engineering important parts of *EDB* volume format which allow them to recover unallocated records. Kaart was also delivered the mapping from internal column identifiers into readable format for some familiar databases in *pim.vol* volumes and created a parser that can automatically extract all allocated records exist in a volume.

2.3 Diversity of Mobile OS

The proliferation of mobile technology has made many companies to produce their own mobile OS. Forensic approaches for Windows Mobile OS might not be applicable to other mobile platforms. Therefore, Savoldi made a brief survey and comparison between mobile forensic for Windows Mobile OS and *Symbian S60* [14]. In his work, Savoldi acquired the evidences using both logical and physical methods. Savoldi was also illustrated the differences and identified possible common methodology for future forensic exploration. Conversely, Mohtasebi was studied four mobile forensic tools; namely *Paraben Device Seizure*, *Oxygen Forensic Suite*, *MIAT*, and *MOBILedit!* to extract evidences from *Nokia E5-00* Symbian phone [15]. The comparison was to check the ability to extract evidence and to examine information types such as call logs, map history, and user data files. On the contrary, Casey was proposed a methodology for acquiring and examining forensic duplicates of user and system partitions; from a device running on *webOS* [16]. These captured data is in *.db3* format and can be analysed using *SQL viewer*. Some information are stored in UNIX string format, seen in date column in the database.

2.4 Bundle Software Package

Most of the mobile manufacturers has provide a software package to communicate with their own products. One of the examples is *Microsoft ActiveSync* which is used for Windows Mobile OS. As for *Apple iOS*, Husain was used *iTunes* to force backup the *iPhone* and logical copy of backup can be found in computer hard drive [17]. This method was able to capture entire data from iPhone without *Jailbreak* the devices. Husain was used *MobileSyncBrowser* to analyze the backup file which is in binary format into list and database. Furthermore, *SQLite Database Browser* was used to analyze database file and *Plist Editor* was used to analyze Apple Property List file. Husain was also able to obtain data such as voice communication, text communication, audio-visual, location information, user activity and online activity related evidence. Similarly, Chun and Park were used *Samsung Kies* to extract SMS, photo and mobile image from Samsung Galaxy S [18]. This analysis also emphasis on the vulnerability when using free public Wi-Fi.

However, most of the bundle software package that were used to acquire evidences, placing an agent into the mobile devices. This action may alter the stored data in mobile devices such as the last synchronization date and time; or the name of the last computer synchronize with the devices. For this reason, Rehault was proposed a method of using *boot-loader* concept; which is non-rewritable and able to protect the evidences from being altered [19]. Rehault was also proposed an analysis method to process specific files with specific format. The main focus in this research was to obtain registry hives and the *cemail.vol* which contain deleted data. By reconstructing back registry hives, digital evidence such as SMS, MMS as well as email can be obtained and retrieved from *cemail.vol*. An extended work for *boot-loader* concept by Rehault was published by Chen for *Android* [20]. The concept of acquisition evidences is similar but this time was using Secure Digital (*SD*) card. This method was claimed can effectively perform the recovery of any deleted data. Besides that, Kumar was proposed an agent based tool developed for forensically acquiring and analyzing in Windows Mobile OS [21]. This tool is develop based on client server approach, whereby client is installed on desktop PC and server agent is inject into mobile devices before acquisition process. As for analyzing process, this tool was able to display and decode the image created during acquisition. This research also make a comparison between *Paraben's Device Seizure*, *Oxygen's Forensics Tool* as well as *Cellebrite UFED* and claimed to perform better in Windows Mobile OS.

2.5 Other Acquisition Techniques

Apart from using the bundled software packages, the other way to obtain mobile images are by using the *SSH* connection. To use *SSH*, the phones should have *SSH* installed and the data has to be transferred to a remote host via the network; in most cases are using wireless connection. However, this is a lengthy process and very time consuming; it may require up to 20 hours depending on the image size. Alternately, Gómez-Miralles and Arnedo-Moreno were presented a novel approach by using an iPad's camera connection kit attached via USB connection [22]. In order to acquire iPad's image, this approach was greatly reduces the transferring time. On the bad side, *Jailbreak* is required in order to gain full access and it is not considered as a forensically sound manner for forensic investigation. For that reason, Iqbal made a research to obtain an *Apple iOS* image without *Jailbreak* the device and run the acquisition process on the RAM level [23]. *Apple iOS* devices need to reboot and enter the recovery mode before connected to their own developed tools. The imaging process was less than 30 minutes and they were successfully developed an acquisition method that protects the integrity of the collected evidences.

Work by Jonkers was used flasher boxes to acquire data but there are some limitations was observed; such as issues in verifying the data integrity and not really practical in normal investigation [24]. Consequently, removable memory become popular alternative for physical acquisition process. Rossi was demonstrated internal forensic acquisition in mobile devices by using removable memory [25] and this work becomes a stepping stone for the *boot-loader* concept. A tool to obtain the data is stored in a removable memory and the acquisition process is performed locally. In addition, this tool not only performs acquisition process, but also compiles log and marks the data with some one-way hash algorithm to provide data integrity. The test result was divided into three condition of mobile devices and result obtained are different. For that reason, Rossi suggest to maintain the device in the most possible original status. However, some approach does not work for volatile memory. As a result,

Sylve was presented the first methodology and toolset for acquisition of volatile physical memory from Android devices [26]. This method was created a new kernel module for dumping memory and Sylve has further develop a tool to acquire and analyse the data. Sylve was also presented an analysis of kernel structures using newly developed volatility functionality. Similar method was also proposed by Dezfouli using force backup in an isolated folder [27], but this is yet to be implemented. On the contrary, Vidas was proposed a general method of acquiring process for *Android* by using boot modes [28]. This technique reconstructed the recovery partition and associated recovery mode of an *Android* for acquisition purposes. The acquired data has to be in recovery image format. Custom *boot-loader* method has become popular in *Android* because the user is able to get root permission; and able to acquire an image of the flash memory. Another research using *boot-loader* method was conducted by Park [29]. This research was mainly focused on fragmented flash memory due to the increase of flash memory deployment in mobile phones.

The newly acquisition method is the live acquisition. Thing was proposed an automated system in acquiring evidences and claimed that this method consistently achieved 100% evidence acquisition rate for outgoing message and 75.6% to 100% evidence acquisition rate for incoming message [30]. Thing was used *Android* as the test platform and *Message Script Generator*, *UI/Application Exerciser Monkey*, *Chat Bot*, *memgrab* and *Memory Dump Analyzer* (*MDA*) as the forensic tools. Although the acquisition rate is high, this method was only tested by using their own developed chat bot and yet to be tested using commercial Instant Messaging. Another live acquisition research is by Lai. Lai was proposed data acquisition in *Android*; and deliver the data to the Google cloud server in real time [31]. This method really can deliver the intended data, but the integrity of the data is questionable. On the other hand, Canlar was proposed *LiveSD Forensics* to obtain digital evidence from both the Random-Access Memory (*RAM*) and the Electronically Erasable Programmable Read Only Memory (*EEPROM*) of Windows Mobile OS. This research was claimed to generate the smallest memory alteration, thus the integrity of evidences is well preserved [32].

3 ACQUISITION METHODOLOGY

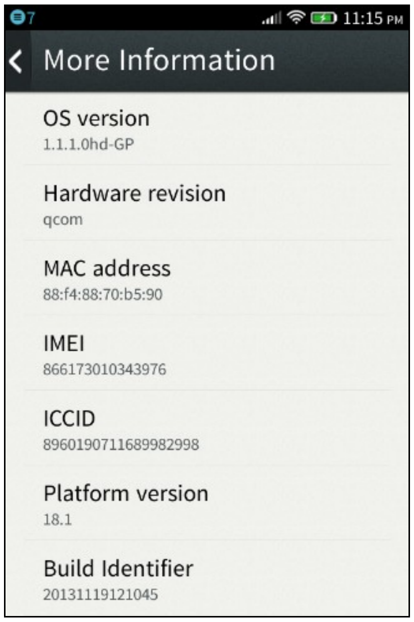
The goal of this paper is to propose a methodology to acquire the mobile forensic image from Firefox OS running phone. In general, Firefox OS architecture consist of 3 layers [33]. The first layer is an application layer called *Gaia* and it works as the user interface for smartphones. The second layer is an open web platform interface; using *Gecko* engine and provide all support for *HTML5*, *JavaScript* as well as *CSS*. All the targeted evidence are stored in this layer. The third layer called *Gonk*; is an infrastructure layer and based on *Linux-Kernel*. There are two types of storage in Firefox OS running phone which are internal storage and additional micro SD card. Acquiring data from the micro SD card is relatively easy; the phone only need to be connected to the host machine and micro SD card can be mounted as removable drive. However, acquiring data from internal storage and other user partitions is quite a challenging tasks. Subsection below will further elaborate about experimental setup and imaging process for Firefox OS running phone.

3.1 Firefox OS Running Phone

For acquisition process, we will use Firefox OS running phone released by *Geeksphone*, model name *Peak*. It was release in April 2013.

Figure 1. Geeksphone Peak

This phone is equipped with Firefox OS version 1.1.1 as shows in Figure 2. Mozilla released an update for their OS regularly and any stable build can be update via over-the-air.



11:15 PM

More Information

OS version
1.1.1.0hd-GP

Hardware revision
qcom

MAC address
88:f4:88:70:b5:90

IMEI
866173010343976

ICCID
89601907116899829998

Platform version
18.1

Build Identifier
20131119121045

Figure 2. Geeksphone Peak Information Detail

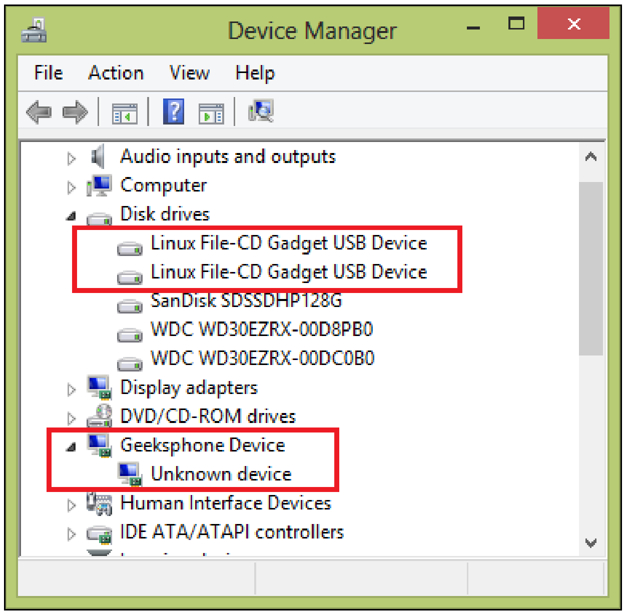
This phone powered by the dual core *Qualcomm Snapdragon S4* processor and based on the *ARMv7* instruction set. Table 1 below shows the specification detail for this phone.

Table 1. Geeksphone Peak Specification

Hardware	Detail
Processor	1.2 GHz Qualcomm Snapdragon S4 8225 processor (ARMv7)
Memory	512 MB Ram
Storage	-Internal 4GB -Micro SD up to 16GB
Battery	- 1800 mAh - micro-USB charging
Data Inputs	Capacitive multi-touch IPS display
Display	540 × 960 px (qHD) capacitive touchscreen, 4.3"
Sensor	-Ambient light sensor -Proximity sensor -Accelerometer
Camera	8 MP (Rear), 2 MP (Front)
Connectivity	-WLAN IEEE 802.11 a/b/g/n -Bluetooth 2.1 +EDR -micro-USB 2.0 -GPS -mini-SIM card -FM receiver
Compatible Network	- GSM 850 / 900 / 1800 / 1900 - HSPA (Tri-band) - HSPA/UMTS 850 / 1900 / 2100
Dimension	-Width: 133.6 millimetres (5.26 in) -Height: 66 millimetres (2.6 in) -Thickness: 8.9 millimetres (0.35 in)

3.2 Forensic Requirement Setup

To begin with a forensic requirement setup, an additional driver for *Geeksphone Peak* need to be installed into the host machine. We will use Windows 8 as an operating system in the host machine. Once connected using the *micro-USB 2.0* port, Windows 8 will ask for the driver. The supported USB driver can be downloaded from *Geeksphone* web. Once the installation finished, *Geeksphone Peak* will appear in the Device Manager as shows in Figure 3 and *micro SD* card will be mounted into the host machine as *Linux File-CD Gadget USB Device*.

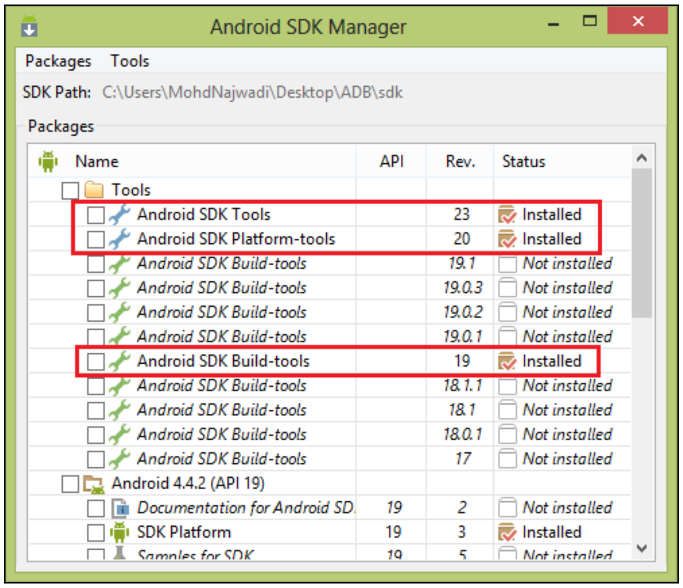


- Audio inputs and outputs
- Computer
- Disk drives
  - Linux File-CD Gadget USB Device
  - Linux File-CD Gadget USB Device
  - SanDisk SDSSDHP128G
  - WDC WD30EZRX-00D8PB0
  - WDC WD30EZRX-00DC0B0
- Display adapters
- DVD/CD-ROM drives
- Geeksphone Device
  - Unknown device
- Human Interface Devices
- IDE ATA/ATAPI controllers

Figure 3. Windows 8 detect the phone as unknown device

Subsequently, we need to make a connection between the phone and the host machine. This connection is important because we need to acquire the data in full image so that can avoid any alteration into the possible evidences. Firefox OS is based on *Linux-Kernel* and the design more or less are similar with *Android*. For that reason, we can easily access the phone using *Android Debug Bridge* (*ADB*). The *ADB* is a toolkit integrated in the Android SDK package and consists of both client and server-side codes. The codes are able to communicate with one another. The bridge connection will be created between the phone and the host machine by using specific port.

Since Firefox OS is a *Linux-based* open-source mobile OS, any rooting procedure are not required. To have *ADB* installed in the host machine, we need to download the `Android SDK` from Android developer page. The SDK file is about 480MB and unzip is necessary. After that, *SDK Manager* is launched and we need to install `Android SDK Tools`, `Android SDK Platform-tools`, and `Android SDK Build-tools` as shows in Figure 4. These three tools are required to run ADB. Now we are ready to start with the acquisition process.



SDK Path: C:\Users\MohdNajwadi\Desktop\ADB\sdk

Packages

Name	API	Rev.	Status
Tools			
Android SDK Tools		23	Installed
Android SDK Platform-tools		20	Installed
Android SDK Build-tools		19.1	Not installed
Android SDK Build-tools		19.0.3	Not installed
Android SDK Build-tools		19.0.2	Not installed
Android SDK Build-tools		19.0.1	Not installed
Android SDK Build-tools		19	Installed
Android SDK Build-tools		18.1.1	Not installed
Android SDK Build-tools		18.1	Not installed
Android SDK Build-tools		18.0.1	Not installed
Android SDK Build-tools		17	Not installed
Android 4.4.2 (API 19)			
Documentation for Android SD	19	2	Not installed
SDK Platform	19	3	Installed
Samples for SDK	19	5	Not installed

Figure 4. ADB Required Tools

3.3 Device Imaging Process

First of all, we need to know what data to be acquired and where it is stored. The targeted data is crucial so that we know what we should acquire and where to proceed. From a forensic standpoint, imaging the whole disk can preserve its contents from any changes. As for Firefox OS, the targeted phone image will consist of several partitions. It will cover the entire systems, user data and installed applications. It also covered several unknown partitions which is need to be further examine.

It is necessary to have the phone battery charged to at least 20%. This is to avoid power loss during acquisition process. Any power loss might lead to device failure and recovery process will become much harder. Furthermore, it is advisable to turn off Bluetooth, 3G data, push notification, push e-mail and localize service to minimize from external interference. Last but not least, we need to unmount micro SD card from the host machine. In order to do that, we need to go to

```
phone Settings > Storage
```

and disable phone storage as shows in Figure 5.

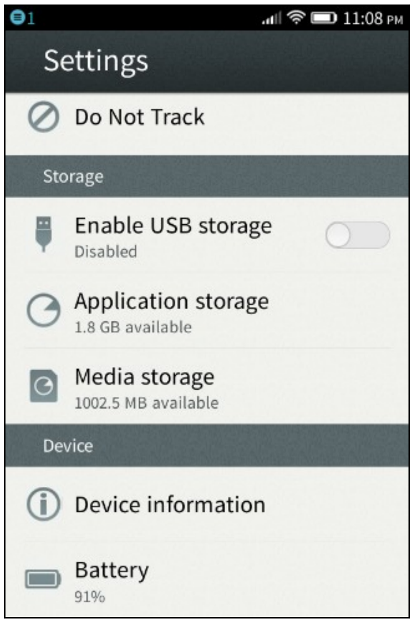
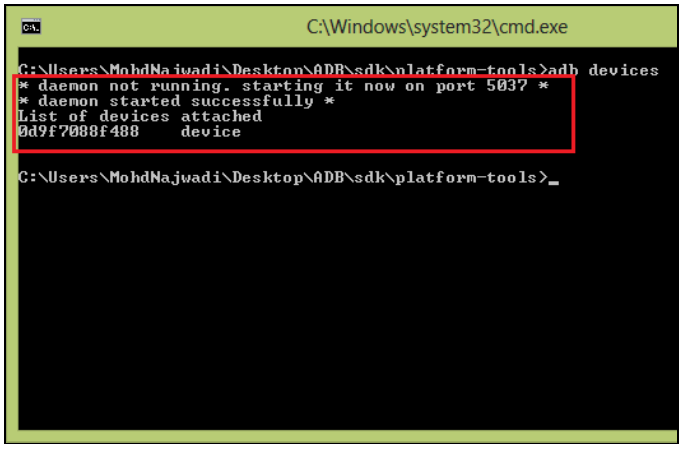


Figure 5. Disable USB Storage

In order to acquire the phone image, we will use UNIX dd command in *ADB* environment. To start *ADB*, we need to run command prompt (*CMD*) and pointing the command start to `%Android SDK%\sdk\platform-tools` folder. After that, we need to check connected phone and type

```
adb devices
```



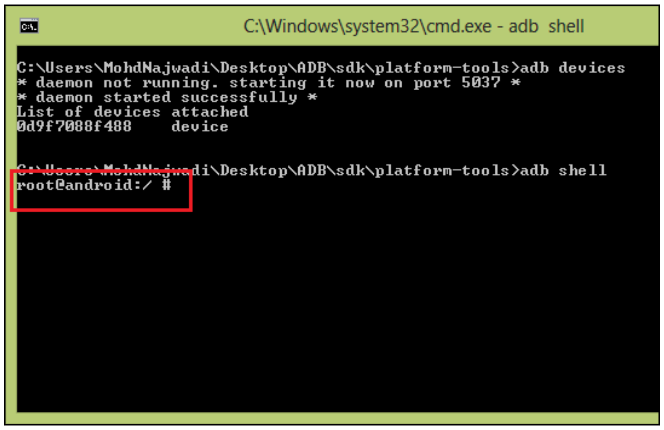
```
C:\Users\MohdNajwadi\Desktop\ADB\sdk\platform-tools>adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
0d9f7088f488    device

C:\Users\MohdNajwadi\Desktop\ADB\sdk\platform-tools>_
```

Figure 6. Attached Devices

During checking, *ADB* will open the connection port and list down all supported devices as shows in Figure 6. Next is to type the following;

```
adb shell
```



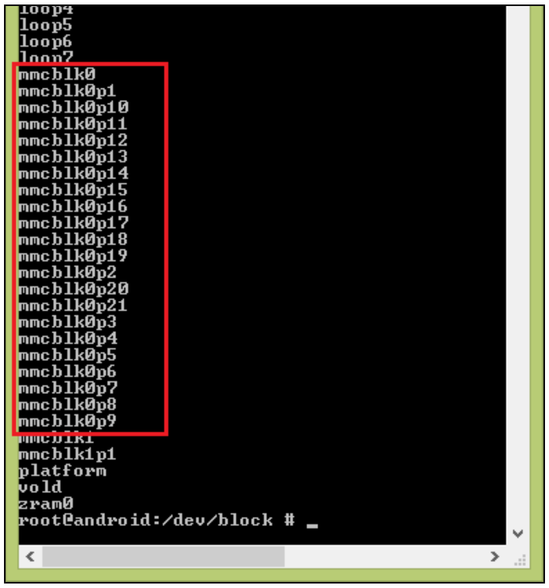
```
C:\Users\MohdNajwadi\Desktop\ADB\sdk\platform-tools>adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
0d9f7088f488    device


C:\Users\MohdNajwadi\Desktop\ADB\sdk\platform-tools>adb shell
root@android:/ #
```

Figure 7. Root Access

This command will establish the connection between the phone and the host machine and `root@android:/ #` access will appear in the *CMD*. In order to check the partition location, type the following;

```
cd dev
cd block
ls
```



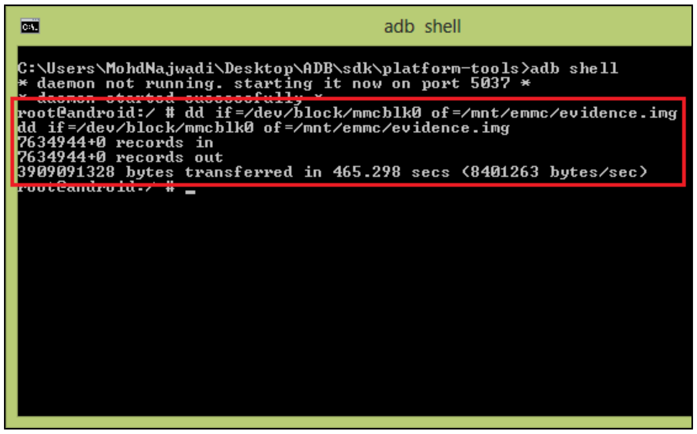
```
loop4
loop5
loop6
loop7
mmcblk0
mmcblk0p1
mmcblk0p10
mmcblk0p11
mmcblk0p12
mmcblk0p13
mmcblk0p14
mmcblk0p15
mmcblk0p16
mmcblk0p17
mmcblk0p18
mmcblk0p19
mmcblk0p2
mmcblk0p20
mmcblk0p21
mmcblk0p3
mmcblk0p4
mmcblk0p5
mmcblk0p6
mmcblk0p7
mmcblk0p8
mmcblk0p9
mmcblk1
mmcblk1p1
platform
vold
zram0
root@android:/dev/block # _
```

Figure 7. List of Partitions

These command will display all the existing partitions in the phone. The targeted phone partition name started with `mmcblk0p1` till `mmcblk0p21`. We have two options to create the phone image. The first option is to run UNIX dd command for each partition, one by one and bit by bit; started with `mmcblk0p1` till `mmcblk0p21`. The second option is to run UNIX dd command to the parent tree of the partition which is `mmcblk0`; and it will later combined all the partitions into one image. We decided to run the second option and type the following;

```
    dd if=/dev/block/mmcblk0
    of=/mnt/emmc/evidence.img
```

The image is pointing into the *micro SD* card and we does not put any block size, by default it is 512KB. The process will take up until 10 minutes depending the block size chosen and the acquired image is around 3.64GB (internal storage size). Figure 8 shows the created log.



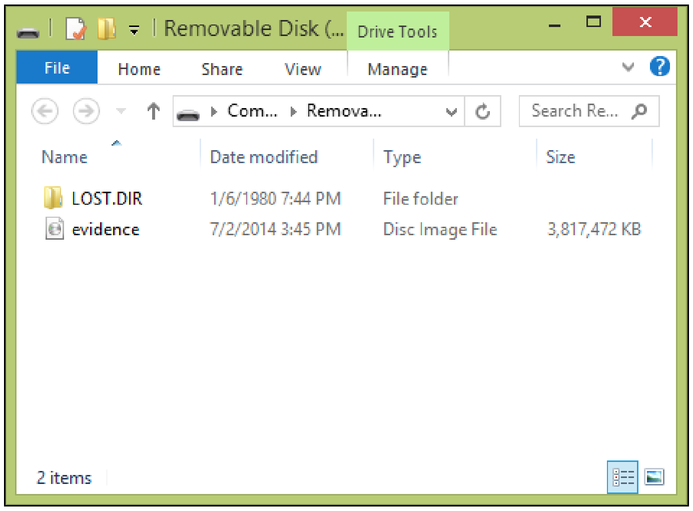
```
C:\Users\MohdNajwadi\Desktop\ADB\sdk\platform-tools>adb shell
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
root@android:/ # dd if=/dev/block/mmcblk0 of=/mnt/emmc/evidence.img
dd if=/dev/block/mmcblk0 of=/mnt/emmc/evidence.img
7634944+0 records in
7634944+0 records out
3909091328 bytes transferred in 465.298 secs (8401263 bytes/sec)
root@android:/ # _
```

Figure 8. Log for Acquired Image

According to Figure 8, the captured time was *465.298 seconds* and the speed was *8.012MB/s.* This acquired image will cover all partitions in the internal memory and also cover unallocated partition. In order to transfer the acquired image into the host machine, we need to mount back the *micro SD* card and follow the previous step in Figure 5 which is

```
phone Settings > Storage
```

and enable back the phone storage. The host machine will again detecting the removable drive and acquired image will be appear as shows in the Figure 9.



Name	Date modified	Type	Size
LOST.DIR	1/6/1980 7:44 PM	File folder	
evidence	7/2/2014 3:45 PM	Disc Image File	3,817,472 KB

Figure 9. Acquired image from the Firefox OS running phone

4 PERFORMANCE MEASUREMENT

The next step is a performance measurement during acquisition process. The test parameter in our measurement is the block size (bs) and the time taken for each acquisition process. In order to obtain an optimal value, we run a series of test for each value and the phone is keep to a minimum activity as possible. This is very important to get the best average result without any external interferences. Under these circumstances, we run three complete 3.64GB dumps for 12 different block size. The speed obtained in each test is shows in Table 2.

Table 2. Throughput obtained with different block sizes

bs	S1 (MB/s)	S2 (MB/s)	S3 (MB/s)	S (MB/s)
32 KB	2.861	2.861	2.861	2.861
64 KB	4.722	4.723	4.723	4.723
128 KB	6.845	6.846	6.845	6.845
256 KB	7.512	7.512	7.512	7.512
512 KB	8.011	8.012	8.012	8.012
1 MB	8.162	8.163	8.162	8.162
2 MB	8.464	8.464	8.465	8.464
4 MB	8.565	8.565	8.565	8.565
8 MB	8.598	8.597	8.598	8.598
16 MB	8.636	8.636	8.635	8.636
32 MB	8.544	8.543	8.544	8.544
64 MB	8.480	8.479	8.48	8.480

Based on this result, there are exponential increase in speed from 32KB to 512KB block size. However the speed differences become smaller when the block sizes used is larger. Figure 10 shows throughput analysis with different block sizes.

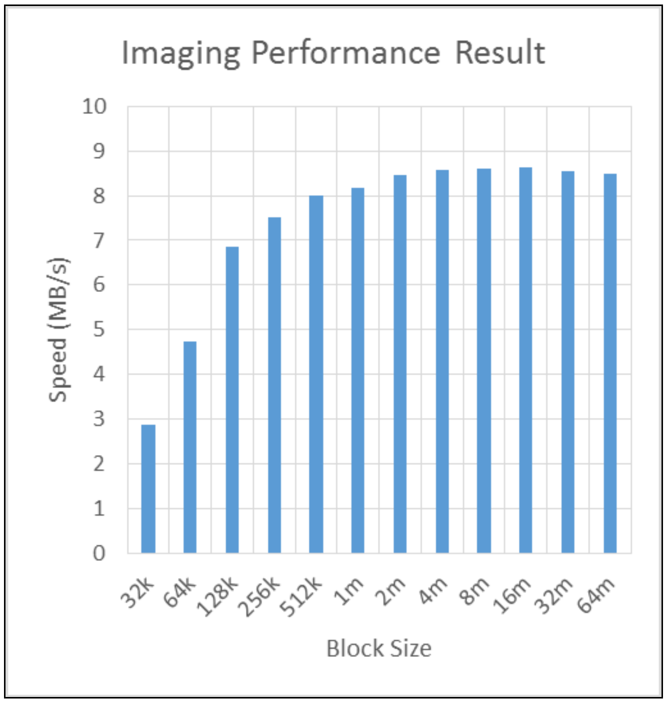


Figure 10. Throughput analysis with different block sizes

For this method we found that we can achieved the best acquisition speed by using 16MB block size. The speed will decrease when the larger block size is used, for this case is 32MB and 64MB block size. On the other hand, the different between 2MB to 64MB block size is too small and not noticeable.

5 CONCLUSION AND FUTURE WORK

The arrival of Firefox OS has created new challenges, concentrations and opportunities for digital investigators. It is a very exciting tasks to explore new version of mobile OS since all the apps is purely built on *HTML5*. To our concern, even though Firefox OS based on Linux-kernel mobile OS, it does not mean existing mobile forensic tools will work fine with this release. We have proved that, only certain data able to read or captured from Firefox OS running phone by using existing mobile forensic tools. This may be due to the differences of existing user data and the partition arrangement. As for performance measurement, we found that the best optimum speed are gained from 16MB block size and the speed will be reduce in the higher block size. As an outcome, we would like to suggest any block size between 2MB to 16MB for optimum acquisition speed. There are many more aspects to be explored. Our next focus will be on analyzing parts and we will go deeper on the system files, user data and application logs.

Acknowledgments. Special thanks to academic staff of Universiti Putra Malaysia for providing continuous guide and support, and also to Ministry of Education Malaysia and Universiti Sains Malaysia for granting the scholarship to me.

6 REFERENCES

1. Mozilla Developer Network, Firefox OS, https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS
2. Mozilla's Boot 2 Gecko and why it could change the world, http://www.knowyourmobile.com/products/16409/mozillas-boot-2-gecko-and-why-it-could-change-world
3. Barmpatsalou, K., Damopoulos, D., Kambourakis, G.: A critical review of 7 years of Mobile Device Forensics, In: Digit. Investig., vol. 10, no. 4, pp. 323--349, (2013)
4. Grispos, G., Storer, T., Glisson, W. B.: A comparison of forensic evidence recovery techniques for a windows mobile smart phone, In: Digit. Investig., vol. 8, no. 1, pp. 23--36, Jul. (2011)
5. Jansen W., Ayers, R.: Guidelines on Cell Phone Forensics - Recommendations of the National Institute of Standards and Technology (2007)
6. Goode, A. J.: Forensic extraction of electronic evidence from GSM mobile phones, In: IEE Seminar on Secure GSM and Beyond: End to End Security for Mobile Communications, pp. 9/1--9/6 (2003)
7. Willassen, S. Y.: Forensics and the GSM mobile telephone system, In: Int. J. Digit. Evid., vol. 2, no. 1, pp. 1--17, (2003)
8. Casadei, F., Savoldi, A., Gubian, P.: SIMbrush: an open source tool for GSM and UMTS forensics analysis, In: First International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE'05), pp. 105--119 (2005)
9. Marturana, P., Me, G., Berte, R., Tacconi, S.: A Quantitative Approach to Triaging in Mobile Forensics, In: 10th International Conference on Trust, Security and

Privacy in Computing and Communications, pp. 582--588 (2011)

10. Chen, S., Hao, X., Luo, M.: Research of Mobile Forensic Software System Based on Windows Mobile, In: 2009 International Conference on Wireless Networks and Information Systems, pp. 366--369 (2009)
11. Irwin, D., Hunt, R.: Forensic information acquisition in mobile networks, In: 2009 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 163--168 (2009)
12. Kaart, M., Klaver, C., van Baar, R. B.: Forensic access to Windows Mobile pim.vol and other Embedded Database (EDB) volumes, In: Digit. Investig., vol. 9, no. 3--4, pp. 170--192, Feb. (2013)
13. Casey, E., Bann, M., Doyle, J.: Introduction to Windows Mobile Forensics, In: Digit. Investig., vol. 6, no. 3--4, pp. 136--146, May (2010)
14. Savoldi, A., Gubian, P., Echizen, I.: A Comparison between Windows Mobile and Symbian S60 Embedded Forensics, In: Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 546--550 (2009)
15. Mohtasebi, S., Dehghantanha, A., Broujerdi, H. G.: Smartphone Forensics : A Case Study with Nokia E5-00 Mobile Phone, In: Int. J. Digit. Inf. Wirel. Commun., vol. 1, no. 3, pp. 651--655 (2012)
16. Casey, E., Cheval, A., Lee, J. Y., Oxley, D., Song, Y. J.: Forensic acquisition and analysis of palm webOS on mobile devices, In: Digit. Investig., vol. 8, no. 1, pp. 37--47, Jul. (2011)
17. Husain, M. I., Baggili, I., Sridhar, R.: A Simple Cost-Effective Framework for iPhone, In: Lect. Notes Inst. Comput. Sci. Soc. Informatics Telecommun. Eng. - Digit. Forensics Cyber Crime, vol. 53, pp. 27--37 (2011)
18. Chun, W., Park, D.: A Study on the Forensic Data Extraction Method for SMS , Photo and Mobile Image of Google Android and Windows Mobile Smart Phone, Commun. Comput. Inf. Sci. - Converg. Hybrid Inf. Technol., vol. 310, pp. 654--663 (2012)
19. Rehault, F.: Windows mobile advanced forensics: An alternative to existing tools, In: Digit. Investig., vol. 7, no. 1--2, pp. 38--47, Oct. (2010)
20. Chen, S.-W., Yang, C.-H., Liu, C.-T.: Design and Implementation of Live SD Acquisition Tool in Android Smart Phone, In: Fifth International Conference on Genetic and Evolutionary Computing, pp. 157--162 (2011)
21. Kumar, S. S., Thomas, B., Thomas, K. L.: An Agent Based Tool for Windows Mobile Forensics, In: Lect. Notes Inst. Comput. Sci. Soc. Informatics Telecommun. Eng., vol. 88, pp. 77--88 (2012)
22. Gómez-Miralles L., Arnedo-Moreno, J.: Versatile iPad forensic acquisition using the Apple Camera Connection Kit, In: Comput. Math. with Appl., vol. 63, no. 2, pp. 544--553, Jan. (2012)
23. Iqbal, B., Iqbal, A., Al Obaidli, H.: A novel method of iDevice (iPhone, iPad, iPod) forensics without jailbreaking, In: 2012 International Conference on Innovations in Information Technology (IIT), pp. 238--243 (2012)
24. Jonkers, K.: The forensic use of mobile phone flasher boxes, In: Digit. Investig., vol. 6, no. 3--4, pp. 168--178, May (2010)
25. Rossi M., Me, G.: Internal forensic acquisition for mobile equipments, In: 2008 IEEE International Symposium on Parallel and Distributed Processing, pp. 1--7 (2008)
26. Sylve, J., Case, A., Marziale, L., Richard, G. G.: Acquisition and analysis of volatile memory from android devices, In: Digit. Investig., vol. 8, no. 3--4, pp. 175--184, Feb. (2012)
27. Dezfouli, F. N., Dehghantanha, A., Mahmoud, R., Binti Mohd Sani, N. F., Bin Shamsuddin, S.: Volatile memory acquisition using backup for forensic investigation, In: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec2012), pp. 186--189 (2012)
28. Vidas, T., Zhang, C., Christin, N.: Toward a general collection methodology for Android devices, In: Digit. Investig., vol. 8, pp. S14--S24, Aug. (2011)
29. Park, J., Chung, H., Lee, S.: Forensic analysis techniques for fragmented flash memory pages in smartphones, In: Digit. Investig., vol. 9, no. 2, pp. 109--118, Nov. (2012)
30. Thing, V. L. L., Ng, K.-Y., Chang, E.-C.: Live memory forensics of mobile phones, In: Digit. Investig., vol. 7, pp. S74--S82, Aug. (2010)
31. Lai, Y., Yang, C., Lin, C., Ahn, T.: Design and Implementation of Mobile Forensic Tool for Android Smart Phone through Cloud Computing, In: Commun. Comput. Inf. Sci. - Converg. Hybrid Inf. Technol., vol. 206, pp. 196--203 (2011)
32. Canlar, E. S., Conti, M., Crispo, B., Di Pietro, R.: Windows Mobile LiveSD Forensics, In: J. Netw. Comput. Appl., vol. 36, no. 2, pp. 677--684, Mar. (2013)
33. Mozilla Developer Network, Firefox OS architecture, https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Platform/Architecture