International Journal of Cyber-Security and Digital Forensics (IJCSDF) 3(4): 183-199
The Society of Digital Information and Wireless Communications, 2014 (ISSN: 2305-0012)

Advances of Mobile Forensic Procedures in Firefox OS

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

ABSTRACT

The advancement of smartphone technology has attracted many companies in developing mobile operating system (OS). Mozilla Corporation recently released Linux-based open source mobile OS, named Firefox OS. The emergence of Firefox OS has created new challenges, concentrations and opportunities for digital investigators. In general, Firefox OS is designed to allow smartphones to communicate directly with *HTML5* applications using *JavaScript* and newly introduced *WebAPI*. However, the used of *JavaScript* in *HTML5* applications and solely no OS restriction might lead to security issues and potential exploits. Therefore, forensic analysis for Firefox OS is urgently needed in order to investigate any criminal intentions. This paper will present an overview and methodology of mobile forensic procedures in forensically sound manner for Firefox OS.

KEYWORDS

Forensic framework, mobile forensic, forensic investigation, forensic methodology, forensic procedures, Firefox OS.

1 INTRODUCTION

Mobile devices are relatively small, portable and widely used by all ages of people in their daily life, business, entertainment, medical, as well as education. Mobile devices consist of mobile phones, smartphones, tablets, and personal digital assistant (*PDA*). The usage of mobile devices are gradually increase over the time especially smartphones and tablets. This increasing trends are due to its useful capability, numerous function and allowed many tasks which required personal computers as well as high processing power to be executed in mobile devices. Figure 1 shows the World-Wide Smartphone Sales (Thousands of Units) classified by mobile OS from Q1 2007 to Q4 2013 [1]. The number of sales are not limited to the smartphone, but also included other mobile devices such as tablets and PDAs. It is because tablets and PDAs using the same mobile OS by smartphone.

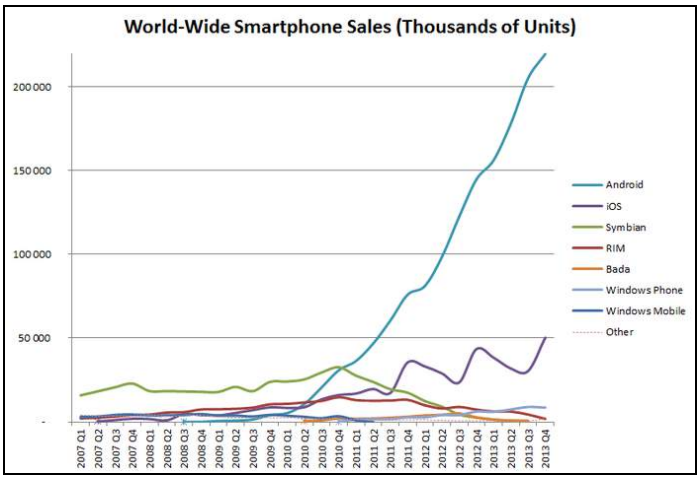

Figure 1. World-Wide Smartphone Sales

Latest analysis by Gartner shows that the total numbers of smartphone sold in Q4 2013 is about 282 million units, while the total numbers of smartphone sold in Q1 2007 is about 24 million units [2-3]. In just 7 years, the total numbers of smartphone sold in Q4 2013 is about 12 times more than the total numbers of smartphone sold in Q1 2007. The highest sales growth goes to *Android* while the second highest goes to *Apple iOS*, a huge gap between first and second place. On the other hand, the remaining mobile OS shows inconsistent sale and sales growth. Figure 2 shows World-Wide Smartphone Sales by percentage [1].

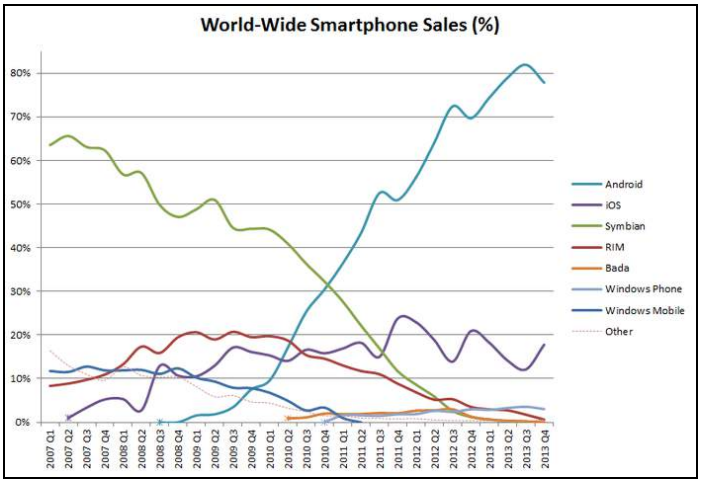

Figure 2. World-Wide Smartphone Sales Percentage

The growth of mobile devices has led to numerous companies to join in the market shares. In Q1 2007, smartphone sales was dominated by *Symbian OS*, followed by Windows Mobile and Research in Motion (*RIM*). However, with the coming of *Apple iOS* in Q2 2007 and *Android* in Q3 2008, domination by *Symbian OS* was slowly reduce. At present, there are no more Windows Mobile as it was replaced by Windows Phone in Q4 2010 and *Samsung Bada* join the race in Q2 2010. In 2014, mobile OS market share is dominated by *Android*, followed by *Apple iOS*, Windows Phone, *RIM*, *Symbian OS*, and *Bada* respectively. In Q1 2012, Mozilla Corporation joined the battle by releasing their own mobile OS, named as Firefox OS [4]. The OS is able to run on selected *Android-compatible* smartphones. The first ever Firefox OS phone was released by *ZTE* in Q3 2013 and followed by *Alcatel*, *LG* and *Geeksphone* [5-6].

Firefox OS is an open source mobile OS which is purely based on *Linux-Kernel* and Mozilla's *Gecko* technology [7]. 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 [7]. Mozilla introduced *WebAPI* by bridging the capability gap between native frameworks and web applications. *WebAPI* will 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* [8].

Unlike *Apple iOS*, Windows Phone, *RIM* and *Android* which full of manufacturer restriction, Firefox OS is based solely on *HTML5*, *JavaScript* as well as *CSS*, and those are totally open sources. By not having any restriction, security issues and potential exploit might come into question. According to Mozilla Developer Network, Firefox OS has designed and implemented multi-layered security model which deliver the best protection against security exploits [9]. In general, Firefox OS is using four layers security model, which are the mobile device itself, *Gonk*, *Gecko* and *Gaia* layers, in order to mitigate exploitation risks at every level. The mobile device is the phone running Firefox OS, while *Gonk* consists of the *Linux-Kernel*, system libraries, firmware, and device drivers. *Gonk* delivers features of the underlying mobile phone hardware directly to the *Gecko* layer. *Gecko* is the application runtime layer that delivers the framework for application execution, and implements the *WebAPI*s to access features in the mobile device. *Gecko* is operating as a gatekeeper that enforces security policies which designed to protect the mobile device from exploitation. *Gecko* also enforces permissions and preventing access of unauthorized requests. Last but not least, *Gaia* is the suite of web applications that delivers user experience [9].

The objective of this paper is to present an overview and methodology of mobile forensic procedures for forensic investigation in Firefox OS. This paper is organized as follows; Section (2) will explain about related work to-date. Section (3) will present the proposed methodology and detail steps in forensic procedure. Section (4) 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 RELATED WORKS

2.1 SIM Cards Investigation

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 [10]. 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 [11]. 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 also can 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.

On the other hand, Casadei was used open source tools, both in Windows and Linux for digital extraction from SIM cards [12]. 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 [13]. 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 Windows Mobile

With the arrival of smartphones, focuses are more on the Windows Mobile OS due to its similarity in nature with 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 contacts, call recording, scheduling, and documents from Windows Mobile OS via Bluetooth, Infrared and USB mode using *Microsoft ActiveSync* [14]. *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 [15].

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 [16]. 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 [17]. 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 [18]. *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 [18]. 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.

Microsoft ActiveSync in Windows Mobile OS placing an agent into 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*. On the other hand, Research by Grispos make a comparison of forensic tools using different approaches for acquisition and decode process [20]. According to Grispos, 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 [20]; but this procedure can damage the device while it is being dismantled. 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. 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 [22].

2.3 Symbian OS

Symbian OS is one of the famous mobile OS in the golden age of smartphone. Forensic work by Mokhonoana and Olivier was discussed about the development of an on-phone forensic logical acquisition tool for the *Symbian OS* [23]. Mokhonoana and Olivier was performed UNIX dd command to acquire the image from the phone. They also discussed different methods of retrieving data from a *Symbian OS* consists of manual acquisition, using forensic tools, logical acquisition, physical acquisition and data acquired from service providers. Additionally, Breeuwsma was proposed a low level approach for the forensic examination of flash memories and describes three

low-level data acquisition methods for making full memory copies of flash memory devices [24]. Their work has been identified as pioneer in physical acquisition techniques using chip-off, *JTAG* and pseudo-physical. The work has been tested on *Symbian OS* devices.

On the other hand, Rossi and Me was proposed a new methodology and a tool to obtain the data by using the removable memory [25]. 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. Test result is divided into three condition of mobile devices and result obtained are different. Therefore, Rossi and Me suggest to maintain the device in the most possible original status. Besides that, Dellutri was proposed a novel methodology by applying data reverse-engineering on Symbian devices [26]. The proposed methodology also shows full potential when running on mobile operating systems which data formats are not open or not public. The investigation used Mobile Internal Acquisition Tool (*MIAT*) and run into more than 50 *Symbian OS* devices. Deluttri was able to capture personal data, *Symbian OS* personal data files format, obsolete data and hidden information during forensic process.

Moreover, Yu was proposed a process model for forensic analysis in *Symbian OS*. The process model consists of five stages which are Preparation and Version Identification; Remote Evidence Acquisition; Internal Evidence Acquisition; Analysis and Presentation; and Review. According to Yu, this model can overcome some problems of forensic investigation in *Symbian OS* [27]. Savoldi and Gubain was presented an assessment about specifically forensic acquisition focusing on security features in *Symbian OS* [28]. They gave an overview about OS and file system architecture. According to Savoldi and Gubain, only physical acquisition was able to recover a bitwise copy of the flash memory without bypassing any security mechanisms. On the other hand, Pooters was created a forensic tool called Symbian Memory Imaging Tool (*SMIT*) to be executed on the *Symbian OS* and created linear bitwise copies of the internal flash memory [29]. *SMIT* able to acquire the image of the internal flash memory and copies the images to a removable memory. *SMIT* has been tested on a *Nokia E65*, *E70* and *N78* model.

Thing and Tan was proposed a method to acquire privacy-protected data from smartphones running the latest *Symbian OS v9.4* and smartphones running the prior *Symbian OS v9.3* [30]. They also presented reverse-engineering analysis work on the active and deleted SMS recovery from the on-phone memory of *Symbian OS*. In addition, Thing and Chua was proposed a forensics evidentiary acquisition tool for *Symbian OS* [31]. Their acquisition tool was built to support a low-level bit-by-bit acquisition of the phone's internal flash memory, including the unallocated space. They also conducted an analysis to perform a detail of the fragmentation scenarios in *Symbian OS*. Apart from that, Savoldi made a brief survey and comparison between mobile forensic investigation in Windows Mobile OS and *Symbian S60* [32]. 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 OS* phone [33]. 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.

2.4 Apple iOS

Forensic investigation and examination in *Apple iOS* was practically started by Bader and Baggili. They performed investigation and examination on logical backup of the *iPhone 3GS* by using the *Apple iTunes* backup utility [34]. Significant forensic evidences such as e-mail messages, *SMS*, *MMS*, calendar events, browsing history, locations services, phone contacts, call history as well as voicemail recording was found and can be retrieved using this *iPhone* acquisition method.

Husain later extended the finding by proposed a simple and cost effective framework for *iPhone* forensic analysis [35]. Followed the same approached by Bader and Baggili, iTunes was used to force backup the *iPhone* and logical copy of backup data can be found in computer hard drive. This method can captured the entire data from *iPhone* without *Jailbreak* the devices. *Jailbreak* is a method to release the security features of the *iDevice* and permits a direct connect to the data inside. Husain was used *MobileSyncBrowser* to analyse the backup file which is in binary format; converting them into lists and databases. Furthermore, *SQLite Database Browser* was used to analyse database file and *Plist Editor* was used to analyse *Apple Property List* file.

In the contrary, Husain and Sridhar was focused on Instant Messaging (*IM*) data in *Apple iOS* [36]. This research made an analysis to forecast the potential use of *IM*s that can lead to cyber bully and cyber stalking. Once the data captured, *Paraben Device Seizure*, *Aesco Radio Tactics* and *Wolf Sixth Legion* were used to analyse the data. The output of this analysis were included username, password, buddy list, last login time and conversation together with timestamp. Similarly, Jung was reviewed the types of Social Network Services (*SNS*) that available in *Apple iPhone* and made further studied on acquisition process in *Apple iOS* platform [37]. Jung made an analysis on eight *SNS* in *Apple iOS* which are *Cyworld*, *Me2Day*, *Daum Yozm*, *Twitter*, *Facebook*, *NateOn UC.*, *KakaoTalk* and *MyPeople*. However, this method required root access to the devices. Therefore, Jung *Jailbreak* the device to get full access permission. The examination and analysis of *SNS* continued by Tso [38]. Tso was discussed five most popular mobile *SNS* applications in *Apple iOS* usages such as *Facebook*, *WhatsApp Messenger*, *Skype*, *Windows Live Messenger* and *Viber*. Tso was followed methods by Husain by forcing *Apple iTunes* to make a logical copy of backup data. Tso ran two experiment, the first data acquisition was after applications installation, while the second data acquisition was after applications deletion.

Moreover, Said was carried a research by comparing *Facebook* and *Twitter* in *Apple iOS*, Windows Mobile OS and *RIM BlackBerry* [39]. The aim of this research is to investigate the different types of logical backup and encryption techniques used in three mobile OS. In the same way, Mutawa later on made *SNS* comparison between *Facebook*, *Twitter* and *MySpace* in three different OS which are *Apple iOS*, *Android* and *RIM BlackBerry* [40]. The examination and analysis was started by uploading picture and post a comment from mobile devices using each *SNS* from different platform. The aimed of this analysis is to determine whether activities performed earlier are stored and can be captured from internal mobile devices memory. Similar research was published by Lee and Park by capturing the *SNS* data using different forensic tools [41]. On the other hand, Levinson explore an investigation for third party applications in iOS platform [42]. According to Levinson, most mobile forensic work emphasis on typical mobile telephony data such as contact information, SMS, and voicemail messages. However, there are heaps of third party application might leave forensically relevant artifacts in mobile devices. For investigation purpose, Levinson acquired data from 8 third party application in *Apple iOS* platform. As a result, Levinson found information about user accounts, timestamps, geo-locational references, additional contact information, native files, and various media files. All these data typically stored in plaintext format and can provide relevant information to a forensic investigators.

Additionally, in order to create *Apple iOS* image, we need to install *SSH* server and transfer the *Apple iOS* internal storage via Wi-Fi into desktop computer. However, this approach may require up to 20 hours. Therefore, Gómez-Miralles and Arnedo-Moreno were presented a novel approach by using an iPad's camera connection kit attached via USB connection [43]. This approach was greatly reduces acquisition 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 [44]. *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. Recently, Ariffin was proposed an operational technique that allows digital forensic investigators to recover deleted image files by referring to *Apple iOS* journaling file system [45]. The proposed method was implemented on an *iDevice* that has been *Jailbreak* and used a customized *RAM* disk that was loaded into the device *RAM*. This technique was successfully recover deleted image files from the journal file and was tested on *iPhone 3GS* and *iPhone 4*.

2.5 Google Android

Android becomes a new mobile forensic investigation emphasis due to its strong hold in the market share and currently have the biggest active user using *Android* platform. *Android* was built based on *Linux-Kernel* and user has the ability to rewrite the firmware, *boot-loader* and other root activities. Research by Lessard and Kessler were among the first in *Android* platform [46]. They performed logical acquisition on *HTC Hero* to acquire physical image. UNIX dd command was performed to get an image from `\dev\mtd` directory. The examination was done on flash memory and removable memory by *AccessData Forensic Toolkit (FTK) Imager v2.5.1*. Apart from that, Anti-forensic among the major consideration in *Android* investigation. Research by Distefano was preserved the data from being damaged by using anti-forensics approach through a local paradigm [47]. This approach was exported some critical data which need to be identified in *XML* format and save it into private directory and it is unreachable from any other applications. Albano also worked on anti-forensics to modify and erase, securely and selectively, the digital evidence in *Android* [48]. This technique do not use any cryptographic primitives or make any changes to the file system. Moreover, Azadegan introduce the design and implementation of three novel anti-forensic approaches for data deletion and manipulation on three forensics tools [49]. This research also identify the limitation of current forensic tools flow design.

Quick and Alzaabi was performed logical and physical acquisition on *Sony Xperia 10i* [50]. Logical acquisition was not able to acquire the full size of the file system, while physical acquisition achieved a bitwise acquisition of the flash memory. They also claimed that they has successfully generated a complete copy of the internal *NAND* memory. On the other hand, Sylve was presented the first methodology and toolset for acquisition of volatile physical memory from *Android* devices [51]. 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. On the contrary, Vidas was proposed a general method of acquiring process for *Android* by using boot modes [52]. 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 [53]. This research was mainly focused on fragmented flash memory due to the increase of flash memory deployment in mobile phones. In addition, Son was conducted an evaluation on *Android* Recovery Mode method in terms of data integrity preservation [54]. Son developed an *Android Extractor* to ensure the integrity of acquired data and presented a case study to test their tool's ability. The test was conducted on seven *Samsung* smartphones with rooted *Android* capability and emphasis on *YAFFS2* and *Ext4* files. The results from the use of *JTAG* method was served as a comparison vector to the Recovery Mode.

In the different research perspective, Racioppo and Murthy made a case study about physical and logical forensic acquisition in *HTC Incredible*

[55]. *AccessData Forensic Toolkit (FTK) Imager v3.0.1* was used to make an image from removable SD memory. After that, they root the device and gaining access to the root directory. They were able to create a bitwise copy of the seven *MTD*s presented in `\dev\mtd` directory. Examination was done using *Ubuntu's scalpel* and claimed that the physical acquisition much more effective to discover corrupted or destroyed files. Moreover, Andriotis was proposed a method for acquiring forensic evidences from *Android* smartphones using open source tools [56]. The investigation was able to obtain information regarding the use of the Bluetooth technology and *Wi-Fi* networks on four *Android* running devices. They specified the files and folders to be targeted and highlighted some security problems that might occur by exposing user's passwords in certain cases by the *Android* system. In addition, Mylonas was extended the acquisition work and focusing on phone's sensors as critical evidence sources [57]. They studied the involvement of sensor like accelerometer, proximity sensor, GPS, compasses, etc in forensic investigation procedures which can be used to infer the user's context. However, they found that the sensor data are volatile and not available in post-mortem analysis. For that reason, they developed the tool named Themis to collect suspect's data. Themis consists of two major parts, workstation version and the mobile agent in *Android*. They presented promising result but yet to prove its effectiveness in practice due to the need of bypass Android security mechanism and need to place mobile agent in each phone.

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 [58]. 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 *IM*. 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 [59]. This method really can delivered the intended data, but the integrity of the data is questionable. As for monitoring, Guido used live forensic to remotely monitor and audit malware activity on *Android* devices [60]. It consists of five elements, each one detecting changes in specific parts of the OS such as *boot-loader*, recovery, file system, deleted files and *APK* files. Even many successful detections, they discovered some malware false positive result and inability to detect some deleted entries.

2.6 RIM BlackBerry

Another major player in mobile market share is *BlackBerry*. As for the *BlackBerry*, Fairbanks was presented a framework for an open source *BlackBerry IPD* file forensics tool [61]. Fairbanks was executed a python tool that parses the *IPD* file upon user request for specific resources. That tool is able to capture the messages, contacts, SMS records, memos, call logs, and the task list from an *IPD* file. This work was tested on *BlackBerry 7290*. The result was good but they did not provide enough data to the potential readers. On the other work, Sasidharan and Thomas generated *BlackBerry* image from Blackberry Acquisition and Analysis Tool (*BAAT*), which is injected on the device before acquisition process [62]. The tool also analyse the forensic image and shows phone contents in different file viewers. However, they unable to read and parse the SMS databases because the version of the *BlackBerry JDE* at that time does not supports *API* to access SMS databases from the device. In the recent *BlackBerry* research, Marzougy was presented the logical acquisition of the *.bbb* file produced by a *Blackberry Playbook* tablet [63]. They used the BlackBerry Desktop Management (*BDM*) software to perform the logical acquisition. The extracted information were varying from system information and user data and also failed to retrieve deleted entries. For future work, they plan to run more tests on a wide range of *BlackBerry* models.

3 PROPOSED WORK

In order to run forensic investigation and analysis, we are proposing this methodology to be conducted during the investigation process. It is based on Smith and Petreski approach [64]. Our methodology and approach consist of three procedures. This methodology is a basic approach and purely designed for Firefox OS. There will be many type of files and analysis, thus it is designed to have specific targeted data checklist. The use of this checklist is to identify relevant data align with specific analysis. This data checklist can be updated from time to time.

3.1 Preparation and Preservation Procedure

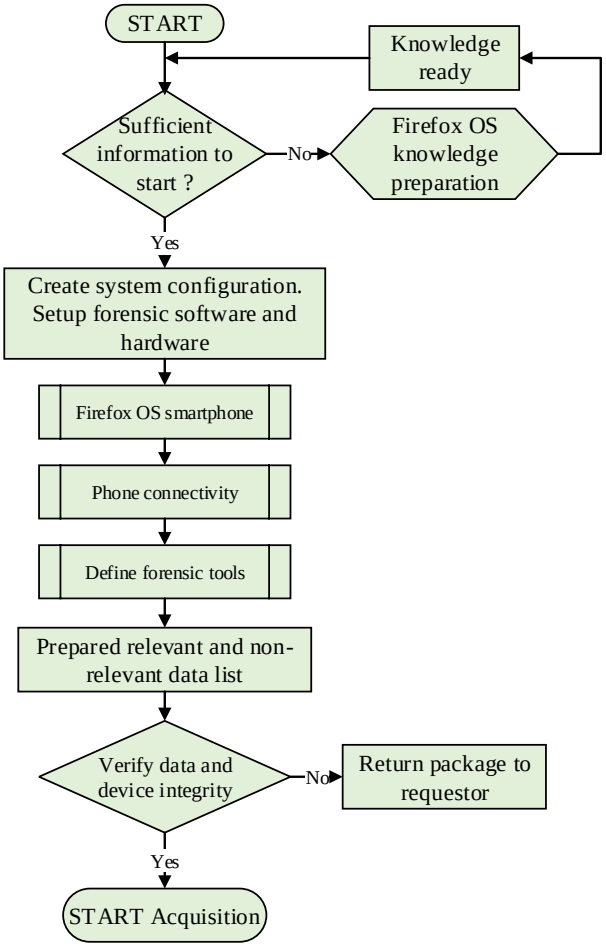

Figure 3. Preparation and preservation procedure

The first procedure is the Preparation and Preservation. This procedure starts with knowledge and information check. If the knowledge about Firefox OS is not sufficient, knowledge gathering is required. It is vital to understand Firefox OS architecture before forensic investigation started. In general, Firefox OS architecture consist of 3 layers [65]. The first layer is an application layer called *Gaia* and work as user interface for smartphones. The second layer is open web platform interface. The second layer used *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 infrastructure layer and consist of *Linux-Kernel*. There are two types of storage in Firefox OS running phone which are internal SD storage and additional *micro-SD* card. Figure 4 shows an illustration version of Firefox OS architecture.

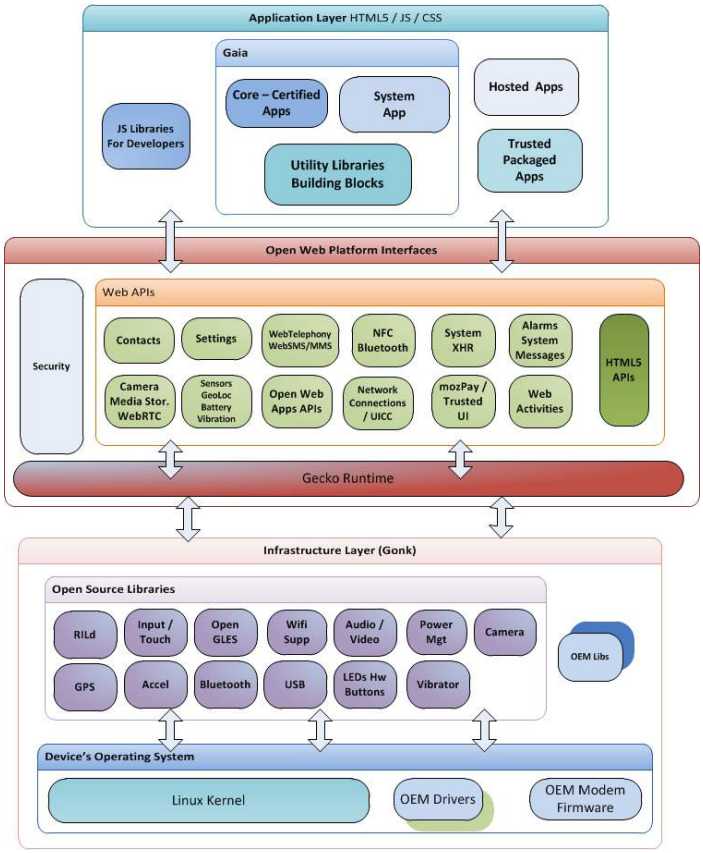

Figure 4. Firefox OS Architecture [65]

The second step is to create a system configuration and setup forensic software as well as related hardware. In this step, we need to have a smartphone preinstalled with Firefox OS, forensic tools, physical and logical connectivity. We will

use *Geeksphone Peak* as our test Firefox OS phone as per shows below

Figure 5. *Geeksphone Peak*

Geeksphone Peak is among the first Firefox running OS phone released and was marketed under developer preview model. It was released in April 2013. This phone is equipped with Firefox OS version 1.1.1 as shows in Figure 6.

Figure 6. *Geeksphone Peak* information detail

Mozilla released an update for their OS regularly and any stable build can be update via over-the-air. 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)

As for the phone connectivity, we are using *micro-USB 2.0* port. Subsequently, we need to make a connection between the phone and the host machine. 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. We will run UNIX dd command from the *ADB* to acquire the phone image. After that, we will use *AccessData Forensic Toolkit* and *HxD Hex Editor* to further examine the acquired image during examination and analysis procedure.

Next we need to prepare a list of relevant and non-relevant data. This list is very important to identify relevant data to be captured later. For example, if we conduct log analysis, relevant data will be chat log, call log, system log and other related information. The next step is to verify the integrity of data and device. Only if the integrity of data and device is confirmed, then we can proceed to Acquisition procedure. Else, the package need to be returned to requestor.

3.2 Acquisition Procedure

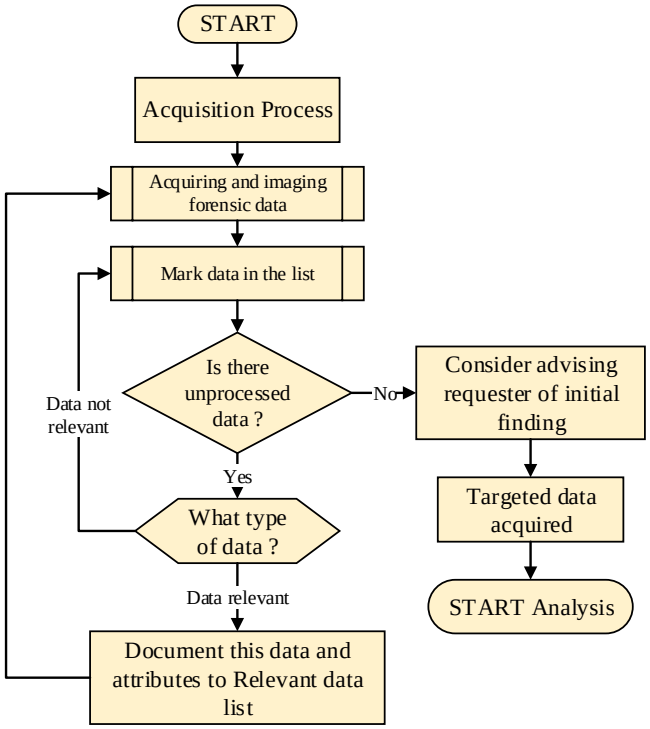

Figure 7. Acquisition Procedure

The second procedure in our proposed methodology is Acquisition Procedure. This procedure is mainly for acquiring and imaging the targeted data. Here, we will use several forensic tools; physical or logical depending on the targeted data to obtain. The process starts with acquiring and imaging targeted forensic data which falls under relevant list. From our observation, we found that the evidences can be captured from internal SD storage, *micro-SD* and other user partitions. Acquiring data from some part of the internal SD storage and *micro-SD* card are relatively easy; the phone only need to be connected to the host machine and internal SD storage as well as *micro-SD* card can be mounted as removable drive. However, acquiring data from other user partitions in internal storage is quite a challenging tasks.

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 8 and internal SD storage as well as *micro-SD* card will be mounted into the host machine as *Linux File-CD Gadget USB Device*.

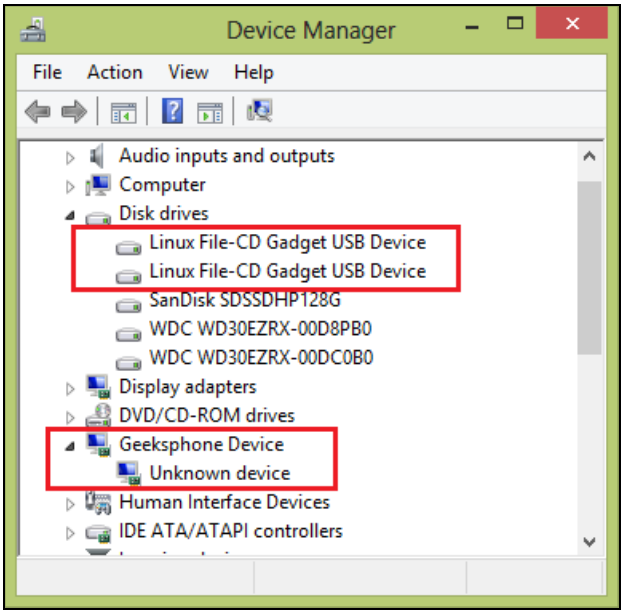

Figure 8. Mounted phone storage

In order to access the storage from the host machine, we need to enable USB storage in the phone setting. To enable the storage, we need to go to

```
phone Settings > Storage
```

and enable USB storage as shows in Figure 9.

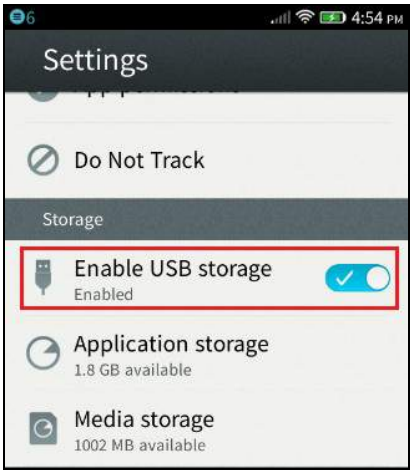

Figure 9. Enable USB storage

After that, we need to go to

`Media storage > Internal Storage`

and enable Share using USB as shows in Figure 10. This option will make an internal SD storage to be appear in the host machine

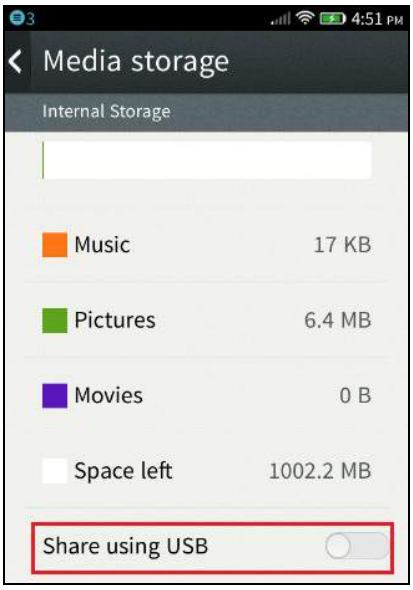

Figure 10. Share internal storage using USB

To access the *micro-SD* card, repeat the same process by go to

`Media storage > SD Card Storage`

and enable it as shows in Figure 11.

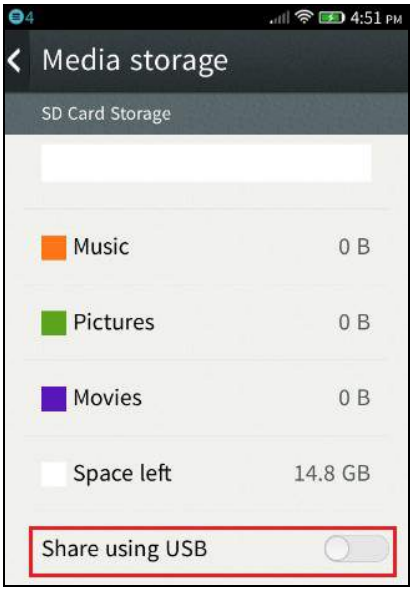

Figure 11. Share *micro-SD* card storage using USB

Now we can see both internal SD storage and *micro-SD* appear in the host machine as shows in Figure 12.

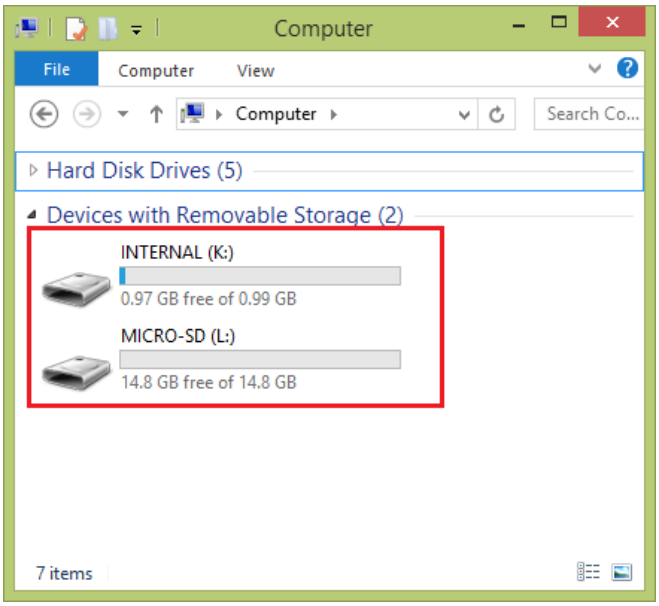

Figure 12. Phone storage mounted as removable storage

From this way, we can access both internal SD storage and *micro-SD* card. However, only around 1GB storage can be access for internal storage, while the remaining 3GB internal storage still cannot be access. Figure 13 shows all the files in the internal SD storage.

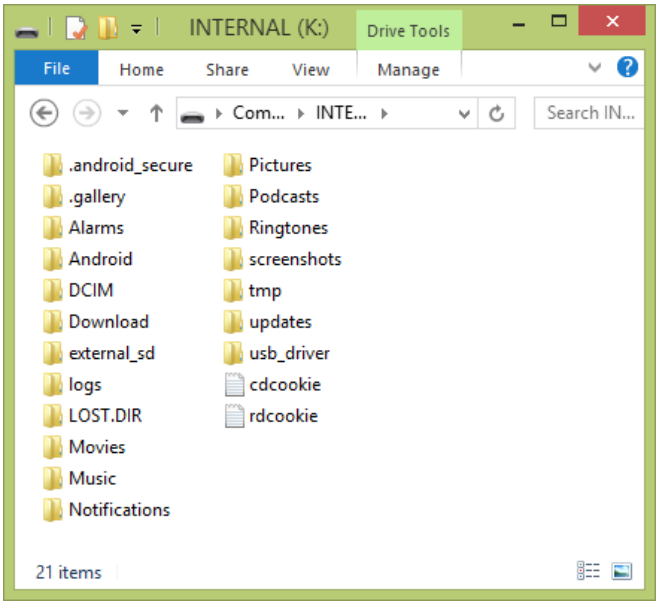

Figure 13. Phone storage

At the time we were conducting this experiment, there is no single file was stored in the *micro-SD* card. As for the internal SD storage, we can simply copy all these files into the host machine and analyse it. For remaining part of the internal storage, we need to run UNIX dd command to acquire the whole image of the internal storage. In order to acquire the phone image, we will use UNIX dd command in *ADB* environment. Before we start the *ADB*, we must disable the USB storage support and unmounts *micro-SD* card from the host machine. After that, we run command prompt (*CMD*) and pointing the command start to `%AndroidSDK%\sdk\platform-tools` folder. To start *ADB*, type the following

```
adb shell
```

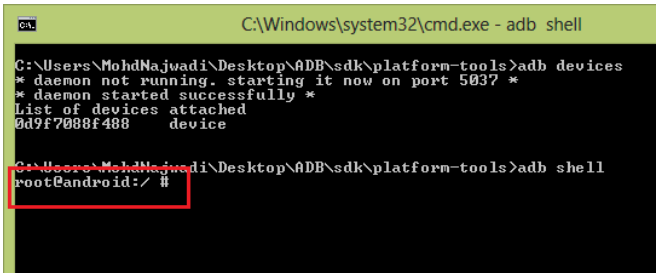

Figure 14. Root Access

This command will establish the connection between the phone and the host machine and `root@android:/ #` access will appear in the *CMD*. After that, we type the following to run dd command;

```
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). This acquired image will cover all partitions in the internal storage 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 9 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 15.

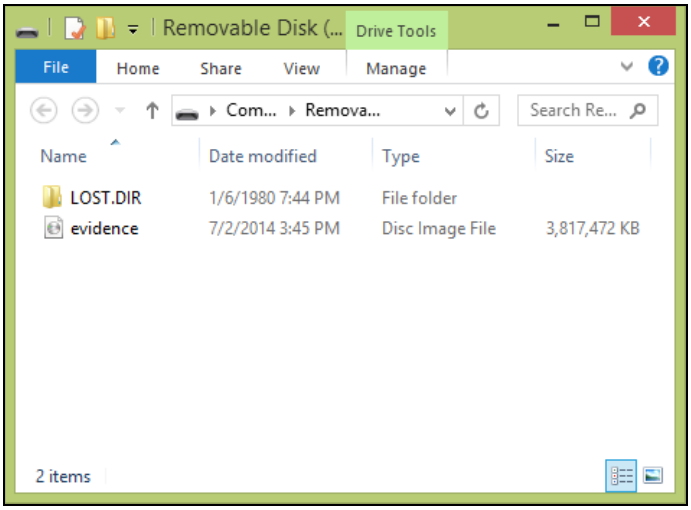

Figure 15. Acquired image from the Firefox OS running phone

After the image is obtained, the relevant data list will be marked and updated. This step only involve specific files in the relevant data list. For example, if we want to run log analysis, only log file will be acquired. The second step is to verify the remaining data, if the remaining data in the smartphone is still relevant, the first process is repeated, and it will write down the details into relevant data list. After all relevant data is obtained, we will gather all initial finding and present it to the requester. Analysis procedure will start after all targeted data acquired.

3.3 Examination and Analysis Procedure

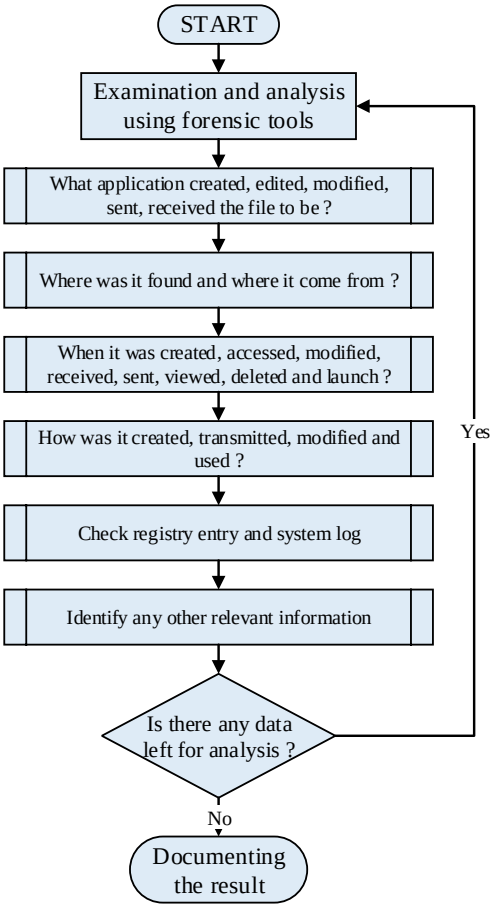

Figure 16. Examination and Analysis procedure

The last procedure in our methodology is Examination and Analysis Procedure. This procedure will focus on deeper attributes of acquired data. All the information will be analysed using additional tools such as *SQL viewer* to open the database file, *HxD Hex Editor* to open unspecified format files or *AccessData Forensic Toolkit* to examine the image file. In this procedure, the acquired data will be examined and analysed to find the application involved in creating, editing, modifying, sending and receiving targeted file. Later, we will investigate the data origin; and the directory it came from. The third sub step is to find when the file was created, accessed, modified, received, sent, viewed, deleted and launched. The fourth sub step is to analyze how it was created, transmitted, modified and used. Registry entry and system log need to be checked and relevant information will be identified and rectified again. Last but not least, we need to repeat all the step, to ensure that there is no missing data during the analysis. Once completed, we can start documenting the result. As for this testing purposes, we try to find the tree directory of the files in the internal SD storage. Table 2 shows tree directory in internal SD storage.

Table 2. Tree directory in internal SD storage

Evidence	Tree Directory
Thumbnail in photo	%Internal%\.gallery\previews\DCIM \100MZLLA
Alarms setting	%Internal%\Alarms
Installed apps	%Internal%\ Android\data
Photo gallery	%Internal%\ DCIM\100MZLLA
Downloads	%Internal%\ Download
Third party apps	%Internal%\ external_sd\Android\data
Event logs	%Internal%\ logs
Recovery folder	%Internal%\ LOST.DIR
Video files	%Internal%\ Movies
Audio files	%Internal%\Music
Notifications log	%Internal%\Notifications
Received picture	%Internal%\Picture
Podcasts info	%Internal%\Podcasts
Ringtones	%Internal%\Ringtones
Screenshots	%Internal%\screenshots
Temp folder	%Internal%\tmp
Downloaded updates	%Internal%\updates

This file analysis only covered for internal SD storage only. For remaining part of the internal storage, we need to further examine the phone image. For that reason, we use *AccessData Forensic Toolkit* to open the captured image. Once we open it, we can see the image consists of 21 partitions as shows in the Figure 17.

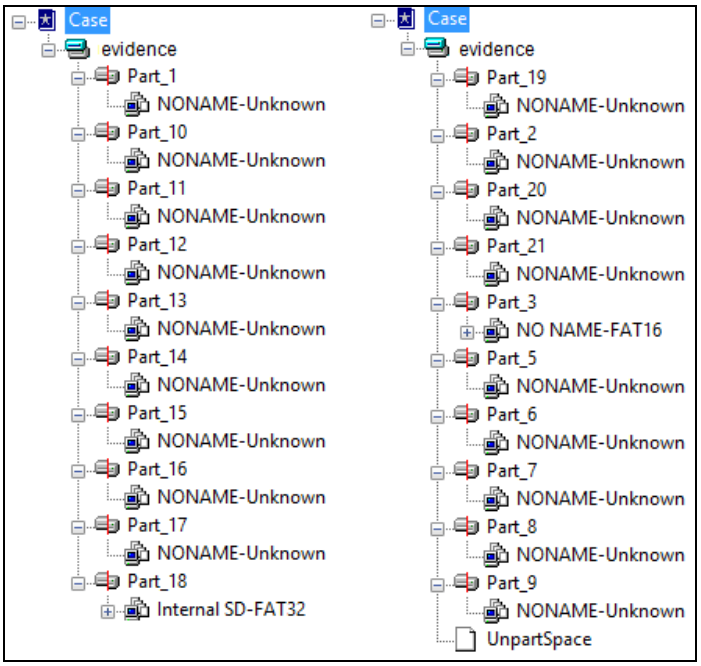

Figure 17. The list of the partition in the phone image

The internal SD storage belongs to partition no 18, named *Internal SD-FAT32*. Other than partition no 3 which is *NONAME-FAT16* and partition no 18, remaining partitions were detected as unknown by *AccessData Forensic Toolkit*. From our observation, we believe that these partition will covers the recovery image, boot partition, system files, local files, cache, user's data and other useful information for forensic investigators.

4 CONCLUSION AND FUTURE WORK

This paper explains about an overview and methodology of mobile forensic procedures in Firefox OS. To our concern, there will be no restriction in Firefox OS and it is design to have the freedom of *HTML5*. Therefore, we will not include any step for rooting the devices; hence the integrity of the data can be preserved. This new approach might be applicable with other mobile platform but it is purely design for Firefox OS. Checklist will be used to classify targeted data during acquisition. Later on, we will work on file, log, system, memory and full data analysis, therefore checklist is very important during each procedures. In this paper we only demonstrated certain steps to show our approach are working perfectly for Firefox OS. Every steps in each procedure has been explained properly to show how we can conduct further experiments.

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.

5 REFERENCES

1. Mobile Operating System Market Share, http://en.wikipedia.org/wiki/Mobile_operating_system#Market_share
2. Gartner says worldwide smartphone sales reached its lowest growth rate with 3.7 per cent increase in fourth quarter of 2008, http://www.gartner.com/it/page.jsp?id=910112
3. Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013, http://www.gartner.com/newsroom/id/2665715
4. First Look at Mozilla's Web Platform for Phones: Boot to Gecko, TechHive, http://www.techhive.com/article/250879/first_look_at_mozilla_s_web_platform_for_phones_boot_to_gecko.html
5. First Firefox OS Smartphone Has Arrived: Telefonica Prices ZTE Open At $90 In Spain, Latin American Markets Coming Soon, TechCrunch, http://techcrunch.com/2013/07/01/first-firefox-os-phone/
6. Mozilla Corporation, Mozilla Announces Global Expansion for Firefox OS, http://blog.mozilla.org/press/2013/02/firefox-os-expansion/
7. Mozilla Developer Network, Firefox OS, https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS
8. 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
9. Mozilla Developer Network, Firefox OS security overview, https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Security/Security_model
10. 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)
11. Willassen, S. Y.: Forensics and the GSM mobile telephone system, In: Int. J. Digit. Evid., vol. 2, no. 1, pp. 1--17, (2003)
12. 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)
13. 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)
14. 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)
15. 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)
16. Klaver, C.: Windows Mobile advanced forensics, In: Digit. Investig., vol. 6, no. 3--4, pp. 147--167, May (2010)
17. Casey, E., Bann, M., Doyle, J.: Introduction to Windows Mobile Forensics, In: Digit. Investig., vol. 6, no. 3--4, pp. 136--146, May (2010)
18 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)
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. 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)
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. 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)
23. Mokhonoana, P. M., Olivier, M. S.: Acquisition of a Symbian Smart phone's Content with an On-Phone Forensic Tool, In: Southern African telecommunication networks and applications conference pp. 1--7, (2007)
24. Breeuwsma, M., Jongh, M. De, Klaver, C., Knijff, R. Van Der, Roeloffs, M.: Forensic Data Recovery from Flash Memory, In: Small Scale Digit. Device Forensics J., vol. 1, no. 1, pp. 1--7, (2007)
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. Dellutri, F., Ottaviani, V., Bocci, D., Italiano, G. F., Me, G.: Data reverse engineering on a smartphone, In: 2009 International Conference on Ultra Modern Telecommunications & Workshops, pp. 1–8, (2009)
27. Yu, X., Jiang, L., Shu, H., Yin, Q., Liu, T.: A Process Model for Forensic Analysis of Symbian, In: Commun. Comput. Inf. Sci. - Adv. Softw. Eng., vol. 59, pp. 86--93, (2009)
28. Savoldi, P. G. Antonio: Issues in Symbian S60 platform forensics, In: J. Commun. Comput., vol. 6, no. 3, (2009)
29. Pooters, I.: Full user data acquisition from Symbian smart phones, In: Digit. Investig., vol. 6, no. 3--4, pp. 125--135, (2010)
30. Thing, V. L. L., Tan, D. J. J.: Symbian Smartphone Forensics and Security: Recovery of Privacy-Protected Deleted Data, In: Lect. Notes Comput. Sci. - Inf. Commun. Secur., vol. 7618, pp. 240--251, (2012)
31. Thing, V. L. L., Chua, T.: Symbian Smartphone Forensics: Linear Bitwise Data Acquisition and Fragmentation Analysis, In: Commun. Comput. Inf. Sci. - Comput. Appl. Secur. Control Syst. Eng., vol. 339, pp. 62--69, (2012)
32. 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)
33. 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)
34. Bader, M., Baggili, I.: iPhone 3GS Forensics : Logical Analysis Using Apple iTunes Backup Utility, In: Small Scale Digit. Device Forensics J., vol. 4, no. 1, pp. 1--15, (2010)
35. 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)
36. Husain, M. I. Sridhar, R.: iForensics : Forensic Analysis of Instant Messaging on, In: Lect. Notes Inst. Comput. Sci. Soc. Informatics Telecommun. Eng. - Digit. Forensics Cyber Crime, vol. 31, pp. 9--18, (2010)
37. Jung, J., Jeong, C., Byun, K., Lee, S.: Sensitive Privacy Data Acquisition in the iPhone for Digital Forensic Analysis, In: Commun. Comput. Inf. Sci. - Secur. Trust Comput. Data Manag. Appl., vol. 186, pp. 172--186, (2011)
38. Tso, Y.-C., Wang, S.-J., Huang, C.-T., Wang, W.-J.: iPhone social networking for evidence investigations using iTunes forensics, In: Proceedings of the 6th International Conference on Ubiquitous Information Management & Communication - ICUIMC '12, (2012)

39. Said, H., Yousif, A., Humaid, H.: IPhone forensics techniques and crime investigation, In: The 2011 International Conference and Workshop on Current Trends in Information Technology (CTIT 11), pp. 120--125 (2011)
40. Mutawa, N. Al , Baggili, I., Marrington, A.: Forensic analysis of social networking applications on mobile devices, In: Digit. Investig., vol. 9, pp. S24--S33, Aug. (2012)
41. Lee, J., Park, D.: A Study on Evidence Data Collection through iPhone Forensic, In: Commun. Comput. Inf. Sci. - Converg. Hybrid Inf. Technol., vol. 310, pp. 268--276, (2012)
42. Levinson, A., Stackpole, B., Johnson, D.: Third Party Application Forensics on Apple Mobile Devices, 44th Hawaii Int. Conf. Syst. Sci., pp. 1--9, Jan. (2011)
43. 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)
44. 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)
45. Ariffin, A., Orazio, C. D., Choo, K. R., Slay, J.: iOS Forensics : How can we recover deleted image files with timestamp in a forensically sound manner ?, In: Eighth International Conference on Availability, Reliability and Security (ARES), pp. 375–382, (2013)
46. Lessard, J., Kessler, G. C.: Android Forensics : Simplifying Cell Phone Examinations, In: Small Scale Digit. Device Forensics J., vol. 4, no. 1, pp. 1--12, (2010)
47. Distefano, A., Me, G., Pace, F.: Android anti-forensics through a local paradigm, In: Digit. Investig., vol. 7, pp. S83--S94, Aug. (2010)
48. Albano, P., Castiglione, A., Cattaneo, G., Santis, A. De: A Novel Anti-forensics Technique for the Android OS, In: 2011 International Conference on Broadband and Wireless Computing, Communication and Applications, pp. 380–385, (2011)
49. Azadegan, S., Yu, W., Liu, H., Sistani, M., Acharya, S.: Novel Anti-forensics Approaches for Smart Phones, In: 45th Hawaii International Conference on System Sciences, pp. 5424–5431, (2012)
50. Quick, D., Alzaabi, M.: Forensic analysis of the android file system YAFFS2, In: Proceedings of the 9th Australian Digital Forensics Conference, pp. 100–109, (2011)
51. 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)
52. Vidas, T., Zhang, C., Christin, N.: Toward a general collection methodology for Android devices, In: Digit. Investig., vol. 8, pp. S14--S24, Aug. (2011)
53. 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)
54. Son, N., Lee, Y., Kim, D., James, J. I., Lee, S., Lee, K.: A study of user data integrity during acquisition of Android devices, In: Digit. Investig., vol. 10, pp. S3--S11, Aug. (2013)
55. Racioppo, C., Murthy, N.: Android Forensics : A Case Study of the ‘ HTC Incredible ’ Phone, In: Proceedings of Student-Faculty Research Day, pp. 1--8, (2012)
56. Andriotis, P., Oikonomou, G., Tryfonas, T.: Forensic analysis of wireless networking evidence of Android smartphones, In: 2012 IEEE Int. Work. Inf. Forensics Secur., pp. 109--114, Dec. (2012)
57. Mylonas, A., Meletiadis, V., Mitrou, L., Gritzalis, D.: Smartphone sensor data as digital evidence, In: Comput. Secur., vol. 38, no. 2012, pp. 51--75, Oct. (2013)
58. 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)
59. 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)
60. Guido, M., Ondricek, J., Grover, J., Wilburn, D., Nguyen, T., Hunt, A.: Automated identification of installed malicious Android applications, In: Digit. Investig., vol. 10, pp. S96--S104, Aug. (2013)
61. Fairbanks, K., Atreya, K., Owen, H.: BlackBerry IPD parsing for open source forensics, In: IEEE Southeastcon 2009, vol. 01, pp. 195--199, (2009)
62. Sasidharan, S. K., Thomas, K. L.: BlackBerry Forensics : An Agent Based Approach for Database Acquisition, In: Commun. Comput. Inf. Sci. - Adv. Comput. Commun., vol. 190, pp. 552--561, (2011)
63. Marzougy, M. Al, Baggili, I., Marrington, A.: LNICST 114 - BlackBerry PlayBook Backup Forensic Analysis, In: Lect. Notes Inst. Comput. Sci. Soc. Informatics Telecommun. Eng. - Digit. Forensics Cyber Crime, vol. 114, pp. 239--252, (2013)
64. Smith, D. C., Petreski, S.: A New Approach to Digital Forensic Methodology, In: DEFCON, 2007 https://www.defcon.org/images/defcon-18/dc-18-presentations/DSmith/DEFCON-18-Smith-SPM-Digital-Forensic-Methodlogy.pdf
65. Mozilla Developer Network, Firefox OS architecture, https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Platform/Architecture