

# **Eliminating Android Security Threats Arising from Deletion Flaws Using Headless Block Swapping (HBS)**

By

# KNUST

Kwame Ofosuhene Peasah

(Dip. Data Processing, BSc. (Hons) Computer Science, MSc. Ind. Mathematics)

A Thesis submitted to the Department of Computer Science, Faculty of Physical Sciences,  
College of Science, KNUST, in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

## DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma at Kwame Nkrumah University of Science and Technology, Kumasi or any other educational institution, except where due acknowledgement is made in the thesis.

Kwame Ofosuhen Peasah

PG1547513

(Student name, Index and ID)

Signature

Date

Certified by:

Dr. Osei Agyei

(Supervisor)

Signature

Date

Certified by:

Dr. M. Asante

(Supervisor)

Signature

Date

Certified by:

Dr. Ussiph Nagim

Signature

Date

## ABSTRACT

The Android operating system continues to dominate the smartphone market with an estimated 85% market share in 2017 (IDC, 2017) due to a growing number of positive features offered by the open source operating system. Rapid growth and improvements to features and capabilities has made it very strong alternative to larger and less mobile desktop operating systems within computing devices in the domain of business and personal functions. Smart phones have now become an integral part of our lives and a common medium of transfer and storage of highly confidential information and transaction. Thus, accessibility of sensitive information on smartphones have become a growing concern due to the lack of ownership consent or approval for access. The flexibility and universal usage of smartphones now serves as an attack vector for privacy infringement and loss of confidential information by attackers who dedicate time and effort to identify and exploit existing smart phone vulnerabilities. This research prioritizes data persistence after deletion as a major concern regarding smartphones and identifies the vulnerabilities android smart phones possess within that domain. The focus area identified the high risks accompanying smartphone device reselling, repossession and disposal to privacy; should the device fall into the hands of anyone who has the intention of extracting data from it. This study investigated the degree of vulnerability for a number of Android smartphone brands which have high market share in Ghana and the knowledge level of users regarding confidential information safety. The study began with a survey to find out which smartphone brands have the most widespread use in the

country as well as the awareness of the smartphone users of the data deletion flaw that is inherent to the currently prevalent versions of Android. Following that, the study made use of open source forensic tools to assess the vulnerability of the previously identified brands and various Android versions, to ascertain the average percentage of data that can be retrieved from the devices after they have been factory reset. Finally, in order to address the absence of an efficient data deletion methodology, the study proposed a data deletion solution, dubbed Headless Block Swapping, which aims to unobtrusively handle sanitizing the storage device after a file has been deleted, by removing the data record from the physical disk to make it impossible to recover.



# Contents

<b>ABSTRACT</b> .....	ii
<b>LIST OF TABLES</b> .....	viii
<b>LIST OF FIGURES</b> .....	x
<b>LIST OF LISTINGS</b> .....	xii
<b>List of Abbreviations</b> .....	xiii
<b>DEDICATION</b> .....	xv
<b>ACKNOWLEDGEMENT</b> .....	xvi
<b>CHAPTER ONE</b> .....	1
<b>1.0 Introduction</b> .....	1
<b>1.1 Statement of the problem</b> .....	3
<b>1.2 Problem Statement</b> .....	7
<b>1.3 Research Questions</b> .....	8
<b>1.4 Aims and Objectives</b> .....	9
<b>1.5 Significance of Study</b> .....	10
<b>1.6 Project scope</b> .....	10
<b>1.7 Project Motivation</b> .....	11
<b>1.8 Limitations of the project</b> .....	11
<b>1.9 Organization of the thesis</b> .....	12
<b>CHAPTER TWO</b> .....	14
<b>LITERATURE REVIEW</b> .....	14
<b>2.0 Introduction</b> .....	14
<b>2.1 Linux File System Overview</b> .....	15
<b>2.1.1 An Overview of the ext4 File System</b> .....	18
<b>2.1.2 Ext4 Security</b> .....	21
<b>2.2 Android Operating Systems Overview</b> .....	22
<b>2.2.1 Android Operating System Architecture</b> .....	23
<b>2.2.2 Android File Systems</b> .....	27
<b>2.2.3 Hierarchy of the Android File System</b> .....	31

2.2.4 <i>The Android Storage System</i> .....	34
2.3 Digital Forensics and Related Works .....	36
2.3.1 <i>History of Mobile Forensics</i> .....	39
2.3.2 <i>Android Mobile Forensics</i> .....	42
2.4 Digital Forensics Models .....	44
2.4.1 <i>Abstract Digital Forensics Model (ADFM, 2012)</i> .....	44
2.4.2 <i>The Enhanced Digital Investigation Process Model (EDIP)</i> .....	46
2.4.3 <i>Computer Forensics Field Triage Process Model (CFFTPM)</i> .....	49
2.4.4 <i>Generic Computer Forensics Investigation Process Model (GCFIPM)</i> .....	51
2.5 Digital Forensic analysis for Android devices .....	53
2.6 Security Metrics for the Android Ecosystem .....	55
2.7 Emerging security threats for mobile platforms .....	59
2.8 Forensic Analysis of Android Mobile Devices .....	62
2.9 Android Digital Forensics: Data, Extraction and Analysis .....	64
2.10 Vulnerabilities in Android Smartphones .....	66
2.10.1 <i>Vulnerabilities of Factory Reset</i> .....	68
CHAPTER THREE .....	71
METHODOLOGY .....	71
3.0 Introduction .....	71
3.1 Survey Work .....	71
3.1.1 <i>Study design and context</i> .....	71
3.1.2 <i>Sample and Procedure</i> .....	73
3.1.3 <i>Data management and analysis</i> .....	74
3.1.4 <i>Ethical consideration</i> .....	75
3.2 Android Data Persistence Analysis .....	75
3.2.1 <i>Introduction</i> .....	75
3.2.2 <i>Overview of Analysis (Method)</i> .....	76
3.2.3 <i>Image Acquisition Tools</i> .....	77
3.2.4 <i>Hardware and Software Tools</i> .....	77
3.2.5 <i>Image Acquisition Process</i> .....	82
3.2.6 <i>Rooting the Device</i> .....	83

<b>3.2.7 Busybox Installation</b> .....	86
<b>3.2.8 Copying the Device Internal Memory</b> .....	87
<b>3.3 FILE CARVING PROCESS</b> .....	90
<b>3.3.1 Installation of File Carving Tools</b> .....	90
<b>3.3.2 File Extraction and Enumeration with Scalpel and Foremost</b> .....	91
<b>3.3.3 File Enumeration Using Scalpel</b> .....	97
<b>3.3.4 FILE ENUMERATION USING FOREMOST</b> .....	100
<b>3.4.5 Analysis of Data Enumeration Results</b> .....	103
<b>3.5 HEADLESS BLOCK SWAPPING</b> .....	108
<b>3.5.0 Introduction</b> .....	108
<b>3.5.1 Overview of the Proposed Solution</b> .....	108
<b>3.5.2 Headless Block Swapping (HBS) Conceptual Algorithm</b> .....	109
<b>3.5.3 Proposed Components of Headless Block Swapping</b> .....	109
<b>3.5.4 File System Snapshot Module</b> .....	111
<b>3.5.2 Deletion Detection Module</b> .....	120
<b>3.5.3 Decapitation Module</b> .....	122
<b>3.5.4 Block Swapping Module</b> .....	124
<b>3.6 Alternatives to the Headless Block Swapping Process</b> .....	129
<b>3.7 HBS Simulation</b> .....	129
<b>CHAPTER FOUR</b> .....	131
<b>4.1 Analysis of Data Enumeration Results after Applying the Headless Block Swapping Algorithm</b> .....	131
<b>SURVEY WORK</b> .....	137
<b>4.2 Survey Work Results</b> .....	137
<b>4.2.1 Biodata of Respondents</b> .....	137
<b>4.2.2 Previous phone used and mode of disposal</b> .....	138
<b>4.2.3 Android phone users, information accessed and data privacy</b> .....	140
<b>4.2.4 Knowledge and awareness about Android security and data recovery</b> .....	144
<b>4.2.5 Parameters for securing information on Android phones</b> .....	147

4.3 Survey Work Discussion .....	148
CHAPTER FIVE .....	152
5.0 INTRODUCTION .....	152
5.1 Assessment of Data Vulnerability .....	152
5.2 Assessment of Headless Block Swapping .....	153
5.3 Novelty/Contribution to Knowledge .....	155
5.4 Conclusion .....	156
References .....	158



## LIST OF TABLES

Table 2.1: Standard block group layout of ext4 .....	18
Table 3.1- Summary of devices that were used for the data persistence analysis .....	78
Table 3.2: Data enumeration results from image files before the devices were reset, using Scalpel. ....	98
Table 3.3 – Data enumeration results from image files after the devices were reset, using Scalpel. ....	99
Table 3.4 – Data enumeration results from image files before the devices were reset, using Foremost .....	101
Table 3.5 – Data enumeration results from image files after the devices were reset, using Foremost. ....	102
Table 3.6 – Percentage of files recovered by Scalpel .....	105
Table 3.7 – Percentage of files recovered by Foremost .....	106
Table 3.8 – Minimum percentages of files extracted by Scalpel for each brand .....	107
Tale 3.9 – Minimum percentages of files extracted by Foremost for each brand .....	107
Table 4.1: Android devices used to test the HBS algorithm .....	130
Table 4.2: Data Extracted by Scalpel Before Factory Reset .....	131
Table 4.3: Data Extracted by Scalpel After Factory Reset .....	132
Table 4.4: Data Extracted by Foremost Before Factory Reset .....	132
Table 4.5: Data Extracted by Foremost After Factory Reset.....	133
Table 4.6 – Percentage of files recovered by Scalpel on devices running HBS .....	136

Table 4.7 – Percentage of files recovered by Foremost on devices running HBS .....136

Table 5.1: Demographic Characteristics of Respondents .....137

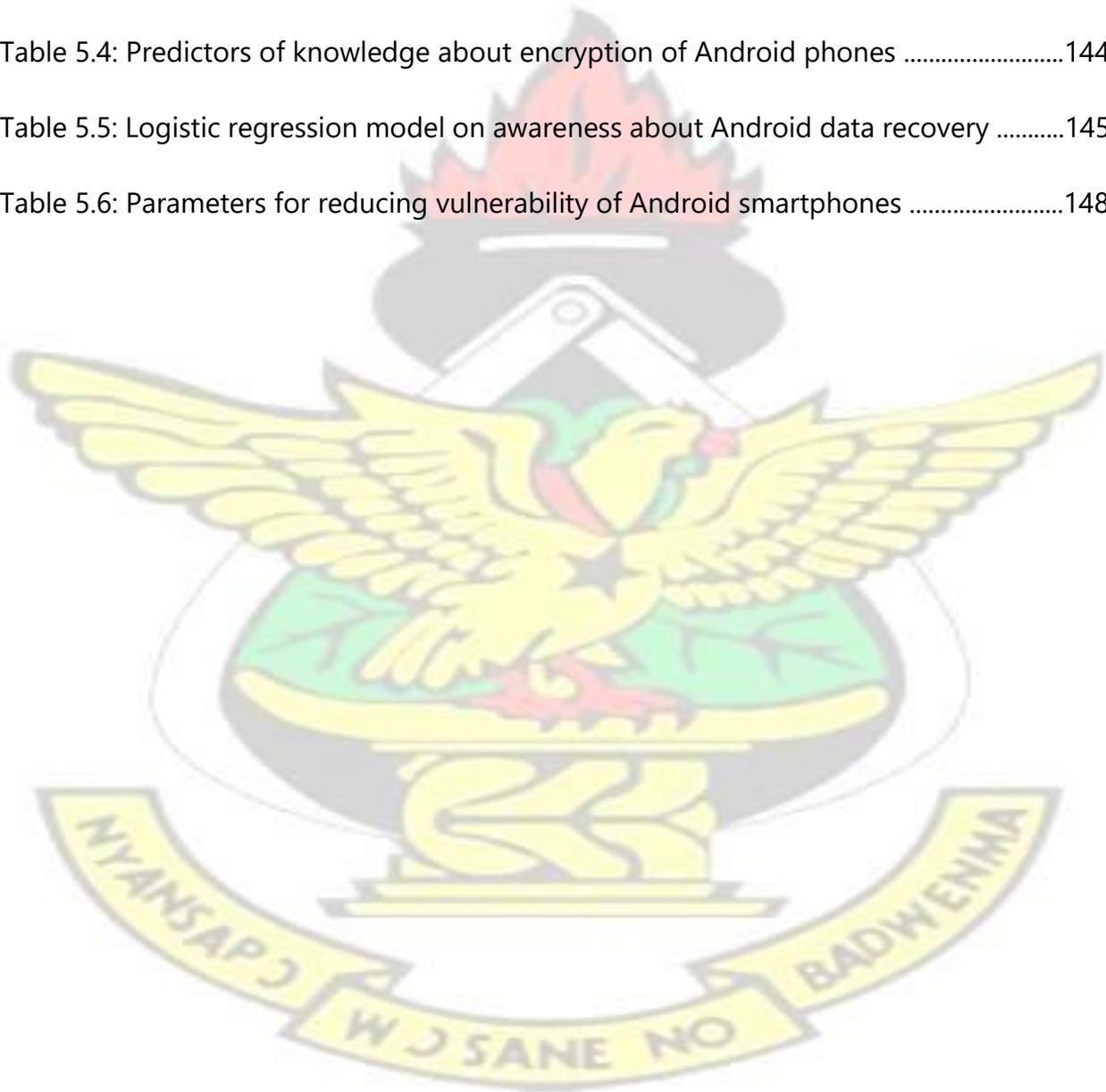
Table 5.2: Chi-square test of previous phone used by biodata of respondents .....139

Table 5.3: Chi-square test on medium of phone disposal and private data protection .142

Table 5.4: Predictors of knowledge about encryption of Android phones .....144

Table 5.5: Logistic regression model on awareness about Android data recovery .....145

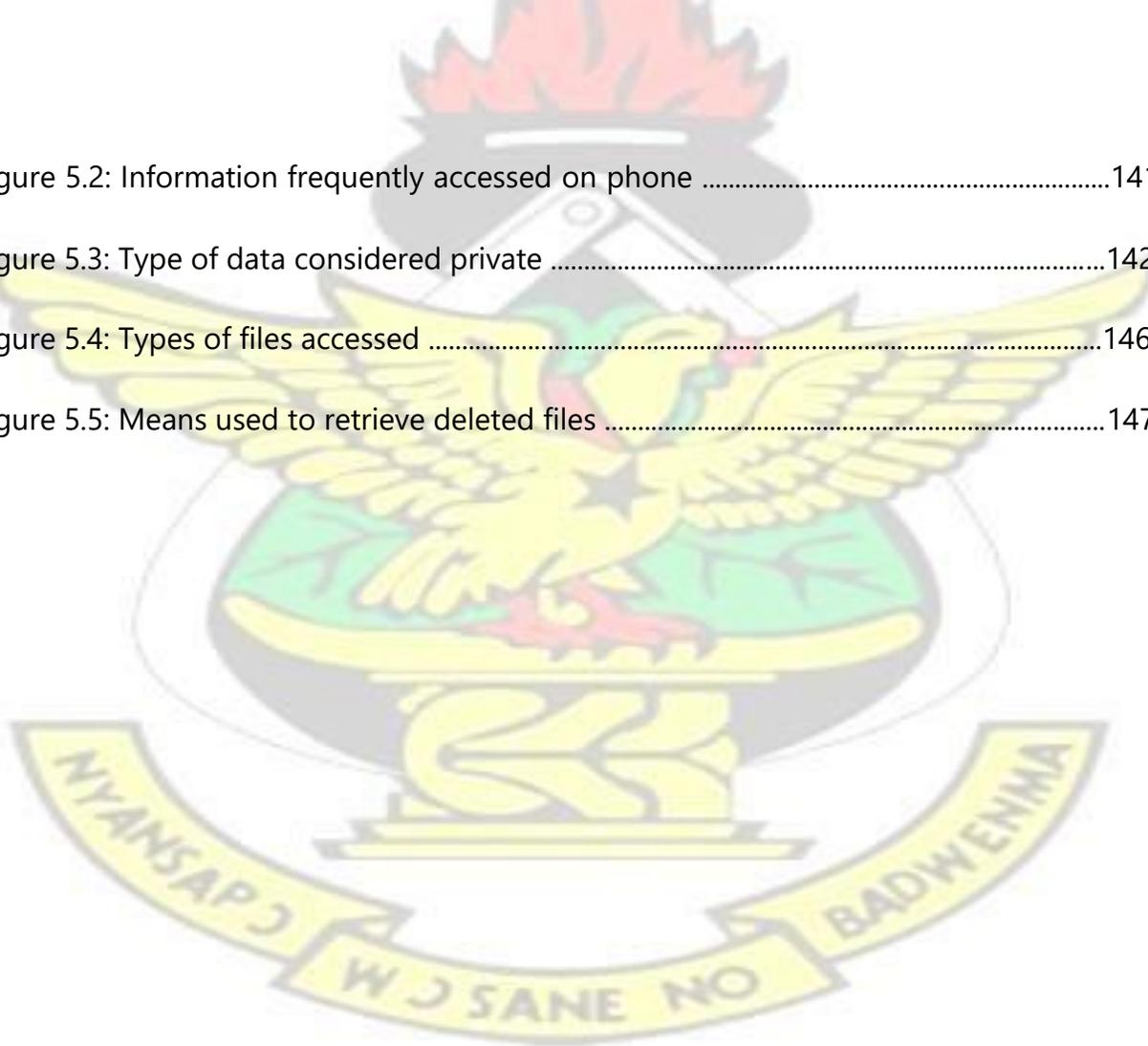
Table 5.6: Parameters for reducing vulnerability of Android smartphones .....148



## LIST OF FIGURES

Figure 1.1 Data lifecycle .....	5
Figure 1.2 Mobile data lifecycle .....	6
Figure 2.2: Android O. S. Architecture (Source: <a href="http://www.tutorialspoint.com/android">www.tutorialspoint.com/android</a> ) .....	25
Figure 3.1: Screen shot of top section of the audit text file from a foremost file carving command .....	95
Figure 3.2: Bottom section of the audit text file from a foremost file carving command	96
Figure 4.1: Headless Block-Swapping .....	110
Figure 4.2: File Snapshot Module .....	119
Figure 4.3: Deletion Detection Module .....	122
Figure 4.4: Decapitation Module .....	124
Figure 4.5: Block Swapping Module .....	125
Figure 4.6: Distribution of Different File Types Extracted by Scalpel Before Factory Reset .....	134
Figure 4.7: Distribution of Different File Types Extracted by Scalpel After Factory Reset .....	134
Figure 4.8: Distribution of Different File Types Extracted by Foremost Before Factory Reset .....	135
Figure 4.9: Distribution of Different File Types Extracted by Foremost After Factory Reset .....	135
Figure 5.1: Mode of disposal of previous phone .....	140

Figure 5.2: Information frequently accessed on phone .....	141
Figure 5.3: Type of data considered private .....	142
Figure 5.4: Types of files accessed .....	146
Figure 5.5: Means used to retrieve deleted files .....	147



## LIST OF LISTINGS

Listing 4.1: shows an algorithm to carry out the FSSM .....	112
Listing 4.2: File System Snapshot Module Pseudocode .....	118
Listing 4.3: Deletion Detection Module.....	120
Listing 4.4: Deletion Detection .....	121
Listing 4.5: Decapitation Module .....	123
Listing 4.5: Data Sanitization (Decapitation and Block Swapping) Module .....	128



CFFTPM	Computer Forensics Field Triage Process Model
GCFIPM	Generic Computer Forensics Investigation Process Model
ADB	Android Debug Bridge ADB
OWASP	Open Web Application Security Project
OSVD	Open source Vulnerability Database
NVD	National Vulnerability Database
CSRF	Cross Site Request Forgery
PASW	Predictive Analytics Software
HBS	Headless Block Swapping



## DEDICATION

I dedicate this study to the LORD Jesus Christ, my beautiful and supportive wife, Mrs Bernice Ofosuhene Peasah, and to my four lovely daughters, Emmanuella Asabea Ofosuhene, Isabella Ofosuhene Peasah, Eirena Asantewaa Ofosuhene Peasah and Edywna Serwaa Ofosuhene Peasah.



## ACKNOWLEDGEMENT

My foremost appreciation goes to God for the strength, knowledge, understanding and wisdom to start and complete this study. Also, I am grateful to my supervisors, Dr Osei Adjei and Dr Michael Asante, for the immense contribution and guidance to the conduct of this study as well as supporting it to the end.

Special appreciation goes to Bishop Dag Heward Mills, the Presiding Bishop of the UDLGOC and Bishops Eddy Addy, Patrick Bruce, Charles Amo and Kakra Baiden of the United

Denomination Lighthouse Group of Churches. Thank you to Mr Joseph Atta Yeboah, Mr Emmanuel Mawuli Abalo and Mr William MorganDarko for the support as well. This work could not have been possible without their support.

Furthermore, special thanks go to Prof Kwesi Obiri Danso, Vice Chancellor of KNUST and the immediate past PROVOST for the College of Science, KNUST, Prof (Mrs.) Ibok Oduro.

Finally, I am thankful to all my colleague lecturers and Staff at KNUST, Department of Computer Science, Prof James Hayfron Acquah, Current Head of Department of Computer Science and Dr Yussif Najim, the former Head of Department of Computer Science.

# KNUST



## CHAPTER ONE

### 1.0 Introduction

Increasingly, mobile devices and for that matter smart phones are, and have become an integral part of our professional and personal lives enabling users to establish connections and interact remotely, irrespective of physical location. Today smart mobile devices appear to be replacing traditional (desktops) and even some contemporary (laptops) technological devices because of the compact and portable size, rich computing resources and large storage capacity of some mobile devices. The popularity of the smartphone dominance is buttressed by the Radicati where mobile users are predicted to exceed 5.6 billion and 6.2 billion in 2014 and 2018 respectively report (Radicati, 2014). However, the mobile device count is estimated to be far above 7.7 billion and 12.1 billion within the same period as mobile users (Radicati, 2015).

Significantly too, the International Data Corporation (IDC) statistical report states that the market share of Android smartphone market alone was 85% in 2017 (International Data Corporation [IDC], 2017). Designed as an open, programmable network device with the capacity to operate as PCs, Android smartphones plays an increasingly important role in personal life and also carries massive classified and private data due to its massive nonvolatile memory (Wilcox, 2009). This notwithstanding, considering the Android OS design flaw in data erasure (Shu *et al.*, 2017), and the subsequent abundance of Android

smartphones (Palmer, 2017; Scrivens and Lin, 2017), users risk being “naked” or vulnerable when sensitive data or information thought to have been deleted falls in the hand of a third party or anonymous user.

However, these emerging technologies though useful is also being employed as attack vectors to commit a large variety of crimes not limited to, fraud, user identity theft/impersonation, unauthorized credit card acquisition and usage (credit card fraud), denial of service attacks, child exploitation/pornography etc (Lee, Palmbach and Miller, 2001). Out of the rising level of danger within the scope of anonymity in computer systems usage as the target or crime object, or as a means/instrument in the crime committed furthermore, as a storage medium for evidence related to the crime. According to (Michael, Mark and Lawrence, 2000) procedures and findings on digital forensics date back to 1984 when FBI and numerous agencies dealing with law enforcement begun to design and implement software to analyze computer evidence. Research community in the likes of the Computer Analysis and Response team (CART), the Scientific Working Group on Digital Evidence(SWGDE), the Technical Working Group on Digital Evidence (TWGDE), and the National Institute of Justice (NIJ) have since been formed with the aim of discussing the discipline with respect to computer forensics with the focus of standardizing the approach to investigations and examination.

Raghavan (2013, p.91) envisaged digital forensics as “the process of employing scientific principles and processes to analyze electronically stored information and determine the sequence of events which led to a particular incident” (Raghavan, 2013). Digital forensics supports investigators in administration of justice by collecting and making available evidence that is valid in law prosecution/court and help prosecutors administer the appropriate sanction pertaining to the case under consideration. Data integrity and the sanctity of evidence in this scenario, digital evidence, is critical to any forensic investigation, thus it is critical to the investigation that the process and means of evidence acquisition follows the rules of engagement in digital forensics. The above concept is supported by (Yusoff, Ismail and Hassan, 2011); adopted process in investigations regarding computer forensics investigation has direct impact on the results. Selecting the wrong investigative process may result in inconclusive results, therefore, resulting in invalid conclusion. Evidences obtained in an unstructured process may result in invalid results that may not be legally feasible in law prosecutions/law court.

### **1.1 Statement of the problem**

Smart mobile computing devices and tablets are one of the most popularly consumed digital devices. Thus, the recent upsurge in the computing power and storage capacity of smartphones as well as other mobile computing devices has made it more convenient to perform most of the things that would otherwise have been done on a desktop or laptop

computer, on the smartphone. Human activities such as banking and finance, health delivery, money transfer, have greatly been influenced positively due to the influx and

proliferation of mobile computing devices which have played host to apps in recent times.

It is therefore commonplace to hear of products such as mobile banking (performing bank transactions on your mobile devices), mobile money transfer, mobile health delivery and the likes due to the convenience afforded by smartphones in terms of portability, significant processing power as well as large and extensible storage capacity.

These mobile technologies enabled transactions or activities leads to data being either transmitted unto those devices or transmitted from them at any point in time. Depending on the data storage structure or file system of the mobile device(s), the data may have its own representation or format coupled with its own encoding and decoding format.

Mobile devices like any other digital consumables have a lifespan or are disposed of at a point in time. The means by which used mobile computing devices are disposed of may include the following;

- Dashing it out as a gift to a relative or comrade
- Swapping the old mobile device for a new one and paying off the difference in the monetary value.
- Recycled or refurbished
- Dismantled to be used as spare parts to service other phones

Any of the above enumerated means of disposing of a mobile device are mostly used

after the user have deleted data which is perceived as important or relevant data or after flashing the mobile device or resetting it to the factory settings. These means of wiping out data is perceived by many users as a secure way of disposing off used mobile devices since there is no visible presence of the existing data.

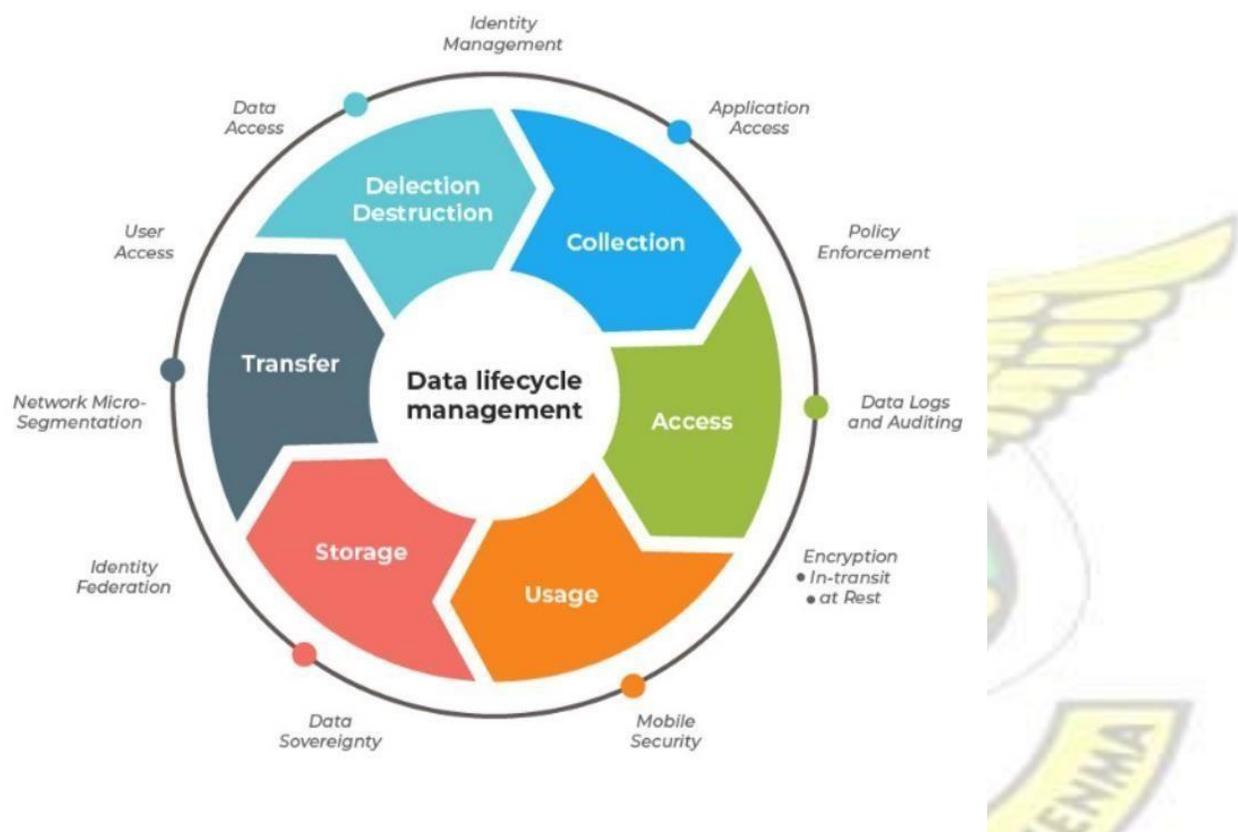


Figure 1.1 Data Lifecycle Management. Source: (Romadhoni, 2019)

The data lifecycle represents the process that contributes to data existence and data nonexistence. It is worth mentioning that, data on mobile devices also have a lifecycle which encompasses data generation/creation, data transformation for storage, usage,

transmission or availability for usage, until its deletion.

Failure to ensure compliance as well as related precautions activities may culminate in corrupting the data, compromising the data or increasing the susceptibility of the data to possible attack(s).

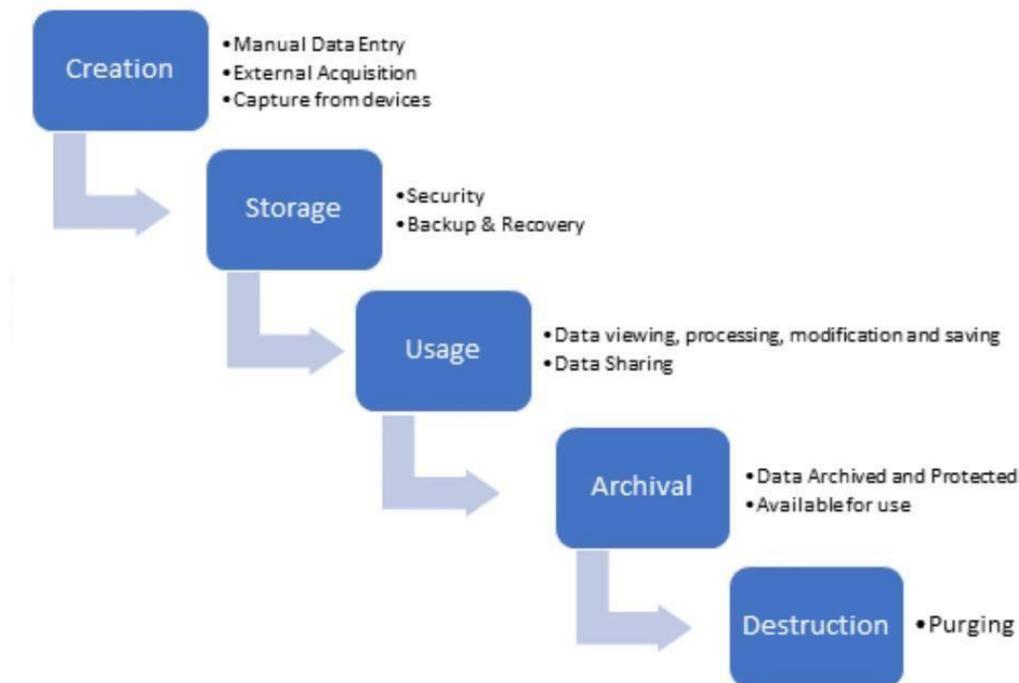


Figure 1.2 Smartphone Data lifecycle.

Perpetrators of digital enabled crimes employ data and perceive used mobile devices as a rich repository of data used to commit diverse crimes not limited to credit card fraud, intellectual property theft, identity theft, etc. In investigations into criminal activities performed using digital devices, a forensic investigation of the suspect's smartphone may very well hold the key to unlocking the case. In addition, several features of smartphone

operating systems and application software allow the history, associations and interactions of the smartphone owner to be traced. This makes the contents of the smartphones of suspects in various criminal activities, a high probative value to forensic investigators. Thus, the intent of deletion, factory resetting as well as flashing of user's mobile device was to conceal sensitive data or information from a third party or anonymous user. However, if the application of these commonly used data erasure methodologies fails to achieve the intent of its usage then the user may appear to be "naked" or vulnerable (susceptible to possible electronic related crime attack).

This project seeks to undertake a survey aimed at measuring the awareness and knowledge about Android smartphones security among smartphone users in Ghana. In addition, this project seeks to examine, visualize and simulate the mobile file system or structure to retrieve traces of digital footprints or data, analyze them so as to ascertain and measure the level of vulnerability of the disposed mobile devices which have been reset to the factory settings, flashed or data deleted. And finally propose modification to the android operating system such that it becomes difficult for perpetrators to recover deleted data from an android device.

## **1.2 Problem Statement**

Several features of the mobile computing devices or the smartphone's operating system and application software allow the history, associations and interactions of the

smartphone owner to be traced. A user may employ some data erasure methodology to delete essential and relevant data from the smartphone without recourse to the data traces, a repository susceptible to possible attack(s).

Android Operating System has a number of data deleting flaws such as data clearing flaws, application uninstallation flaws and factory reset flaws (Shu *et al.*, 2017) which makes it very easy to recover deleted data from an Android device even after a factory reset has been performed. Since most users of Android device make use of only the default deletion and factory reset functions on their devices, most of the data on second hand phones can be recovered by using free and open source forensic tools. Thus, many Android device users stand the risk of having their personal data recovered from devices that they have discarded or sold, even after performing a factory reset.

### **1.3 Research Questions**

This study investigates the various techniques used by smartphone users to make sensitive data inaccessible to third parties such as a forensic analyst. The specific questions that the research seeks to answer are these:

- a. What is the level of awareness and knowledge about Android smartphones security

amongst smartphone users in Ghana?

- b. What methods do Android smartphone users use to hide, delete or wipe (HDW) sensitive data from their smartphones?
- c. By what means do smartphones users employ to dispose of their used phones?
- d. What amount of relevant and critical data resides on a smartphone after its contents have been hidden, deleted or wiped?
- e. What flaws exists in the current Android Operating System with regards to data deletion
- f. What improvements can be introduced to the Android operating system with regards to data deletion?

#### **1.4 Aims and Objectives**

In order to answer the aforementioned questions, the study aims at determining the level of Android data erasure vulnerability and how this vulnerability can be minimized or eliminated with these specific objectives:

1. discover the Android smartphones' knowledge and security awareness and smartphone users in Ghana
2. discover the processes, techniques and tools that smartphone users employ to hide, delete and wipe sensitive data from their smartphones.

3. explore various techniques which forensic analysts can use to acquire data from devices from which the owners have tried to wipe evidence of wrong doing or any other data that is of probative value.
4. demonstrate the Vulnerability of Android Devices after data deletion by extracting data from the devices. i.e obtaining physical image of the eMMC and using data extraction tools (Open source) to extract user data from the disk image.
5. propose a simple yet highly efficient and systematic method for data extraction from smartphones from which data has been hidden, deleted or wiped.
6. design and implement a Headless Block Swapping (HBS) algorithm into the Android kernel to cater for the vulnerability resulting from data deletion.

### **1.5 Significance of Study**

As the major focus of the research work, to help determine, predict and measure the level of vulnerability of used or disposed mobile device. This is very important in view of the high mobile device usage and penetration. Mobile statistics report by the Radicati Group, Inc. stated that the number of mobile users worldwide as at 2014 is over 5.6 billion. It is obvious that, a great population size will be adversely affected in case of any security vulnerability. Again, this research will help to identify a more secure means of disposing one's mobile device to eliminate or reduce the exposure to possible security threat(s).

## **1.6 Project scope**

Mobile computing devices consist of array of devices ranging from Personal Digital Assistants (PDAs), smart phones, Tablet PC and mobile phones. These mobile devices have their own operating system (OS) which runs on them. Notable mobile OS include; Android OS, Windows OS, iOS, Symbian OS and Blackberry OS. However, this project will consider smartphones running Android to undertake the forensic investigation, analyze its file system when new and after it's been used. This will help to know the available data on the device, its relevance and hence its vulnerability of being attacked. Secondly the outcome of the research will help us to know whether a perpetrator of crime can delete every trace of evidence. The research will also review models that are notable in the field of digital forensics for mobile devices, analyze those existing models in order to detect opportunities and flaws directly relating to the models being investigated and develop a detailed framework to aid investigative model selection.

## **1.7 Project Motivation**

The need to secure mobile devices and to minimize level of vulnerability which may be exploited to perpetrate technology related crimes. To provide some level of guidance and directions when undertaking mobile forensic investigations to retrieve timely and relevant data residing on a mobile device.

## 1.8 Limitations of the project

The influx of mobile computing devices or smartphones comes with a commensurate number of mobile operating systems ranging from Symbian, iOS, windows, blackberry OS, Android, etc. The scope of this thesis is limited to smartphones or mobile computing devices which employs Android OS as its underlying mobile operating system.

## 1.9 Organization of the thesis

**Chapter 1** deals focuses on the formulation research problem, overview, it's scope and with some background study as well as the formulation of the research problem, outline of the research questions, the scope and main objectives of the project.

**Chapter 2** reviews literatures related to the aims or objectives of this project work. Literatures reviewed in this chapter covers areas such as Linux file system, Android OS architecture, Android file system, digital forensics as a discipline, digital forensic models, mobile forensics, security threats of the Android OS as well as vulnerabilities of the Android OS in relation to data erasure methodologies. The last part provides a summary of the reviews performed.

**Chapter 3** describes how the study was conducted in three phases, beginning with a survey to understand the permeation of the Android operating system in the Ghanaian smartphone market as well as the brands that have the larger market share in the country.

From there the chapter proceeds to describe how data can be extracted from used

Android devices using open source forensic software tools, both before and after a factory reset has been carried out on the device. The chapter also set out to propose a solution to the data persistence problem. The proposed solution, dubbed "Headless Block Swapping" or "HBS", was conceived as a background process which monitors the file tree to listen for deletions and erase the persistent data from the disk. The chapter concludes with some proposed solutions to the problem of files being recoverable even after a factory reset has been carried out on an Android device.

**Chapter 4** This chapter presents the analysis data enumerated from devices with HBS running on them and the results and discussions from the survey work.

**Chapter 5** chapter presents the results, discussions and conclusion from the survey work.

**Chapter 6** the findings of the study are summarized and discussed with respect to the best ways to secure Android devices from data recovery by malicious third parties who have physical access to the device. The chapter talks about the Headless Block Swapping proposed earlier in the study and discusses how it compares against existing data protection solutions.

# KNUST

## CHAPTER TWO

### LITERATURE REVIEW

#### 2.0 Introduction

This chapter details the review of literatures and related works in line with the objectives of this research work. In view of that, some major thematic areas were reviewed and notable amongst these core areas are but not limited to the following; the Linux file system, Android Operating System, Android File system and hierarchy, Digital Forensics, Mobile Forensics, Digital Forensics investigative models, security of the Android Mobile platform, as well as Android Vulnerabilities. The review of the Linux file system looked at the control of data storage and retrieval. The three segmented parts of the file systems, namely; logical file system, virtual file system and physical file system are considered. The Android operating system built on the Linux kernel was then reviewed by looking at her works of the Open Handset Alliance (OHA), the Android operating system architecture, the Android file system such exFAT, F2FS, JFFS2, etc., the hierarchy of the Android storage and file system.

Related publications and works on digital forensics, some notable digital forensics investigations models were reviewed to ascertain its possible strengths and weakness whilst focusing on mobile forensics as a discipline, with emphasis on the Android mobile forensics. Emerging security threats for mobile platforms were considered and climaxing it with the review of the vulnerabilities in Android smartphones. With respect to this thematic section, vulnerabilities in relation to the amount of the data traces due to factory reset, normal deletion, smartphone flashing, and other data erasure methodologies were studied.

## **2.1 Linux File System Overview**

File systems in computing determines the storage and retrieval of data processes. In the absence of this file system, storage medium content would be seen as a large pool of data, failing to identify the start and end of a sequence of data. By segmenting data and tagging them with an identification property, we can easily isolate and locate information. From the concept of paper-based information system nomenclature; a group of data can be referred to as a "file". The rules and structuring logic used in information management is referred to as "File system". With a large variety of file systems available, varying in logic and structure, speed in relationship to size, security and audit process and many more.

With some file systems purposely designed for applications and others for optical discs (ISO 9660). A large number of storage devices utilize varied file systems based on media

being transmitted or contained. The hard disk drive is the most common storage device in use today with other kinds of storage media still adopted as a means of storing information such as magnetic tapes, optical discs, etc. In the case of computer memory (RAM) temporary file systems are created upon accessing the media and performing operations.

In computing, a *file system* is used to control how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with no way to tell where one piece of information stops and the next begins. By separating the data into pieces and giving each piece a name, the information is easily isolated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file". The structure and logic rules used to manage the groups of information and their names is called a "file system". There are many kinds of file systems. Each one has different structure and logic, properties of speed, flexibility, security, size and more. Some file systems have been designed to be used for specific applications. For example, the ISO 9660 file system is designed specifically for optical discs.

File systems can be used on numerous different types of storage devices that use different kinds of media. The most common storage device in use today is a hard disk drive. Other kinds of media that are used include flash memory, magnetic tapes, and optical discs. In

some cases, such as with tmpfs, the computer's main memory (random-access memory, RAM) is used to create a temporary file system for short-term use.

Some File Systems are a fundamental part of Storage devices (Prabhakaran *et al.*, 2005), providing a unique way of organizing data within the device, which may differ from the file system of the System accessing the device and also providing file access via a network protocol (NFS, SMB). Some file systems are also computed upon request or merely providing a mapping to another different File system that may serve as a backup store. These File systems are called Virtual File Systems. Eg (Procfs). File System manage both access to the content of files or the metadata (Referring to the descriptive information about a file). File Systems are responsible for arranging storage space, reliability efficiency and management of spaces for faster access are mainly dependent on the design of many physical media. Many File Systems are segmented into three parts, but in some cases these three parts are either defined independently or together. The three segmented parts of the file systems are;

**LOGICAL FILE SYSTEM:** Which is responsible for the interaction with the user applications.

Responsible for providing the APPLICATION PROGRAM INTERFACE (API) for many file operations – OPEN, CLOSE, READ, etc. The logical file systems pass request operation to

the layers below for processing. In conclusion, it provides file, directory and security access.

**VIRTUAL FILE SYSTEM;** Is an interface that allows support for multiple concurrent instances of various physical file system, to provide multiple data allocation and addressing at the same time in the cases of multiple applications or processes running concurrently. Each instance is called a File System Implementation (IBM Corporation, 2016).

**PHYSICAL FILE SYSTEM:** Focuses on the physical operation of the storage device. Physical File System process physical blocks / data units being read or written. It handles buffering and memory management and controls the physical allocation of blocks in specific locations in storage on the storage medium (Amir, 2016). This section has a wide range of functionalities with regards to maintaining the integrity and smooth operations of the Device through storage optimization and wiping techniques.

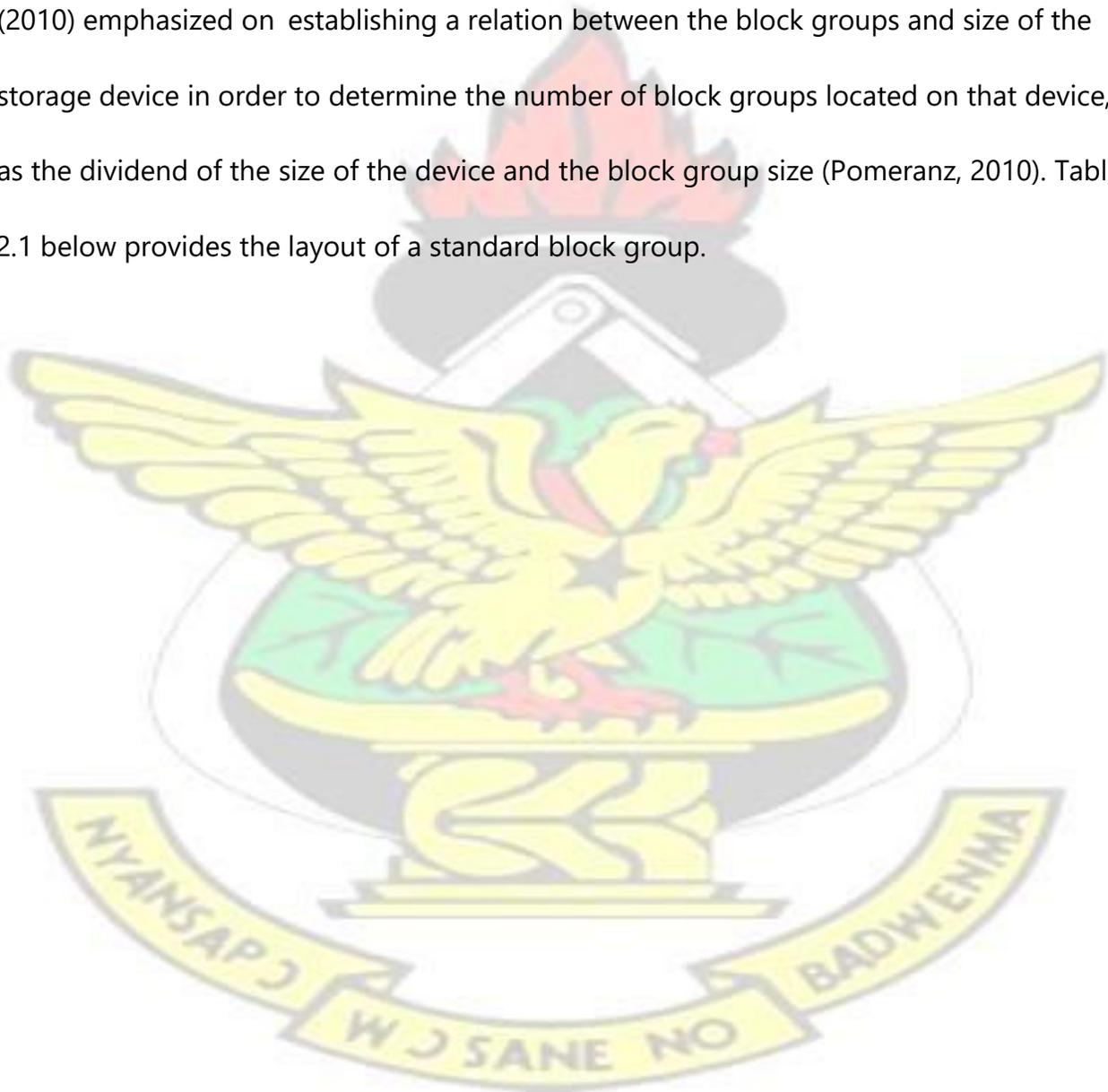
### ***2.1.1 An Overview of the ext4 File System***

The Ext4 File System was a successor to the Ext3 file system with both file systems originating from the MINIX file system, a small, Unix-like operating system for IBM PC/AT microsystems, referred to mainly as a "Journaling file system", focused on eliminating the vulnerabilities and latency delay in read and write operations within the file system. Unlike the improved Ext4 file system, the MINIX file system could only handle 14 characters with a

storage space for address maximizing at 64MB. The Ext4 is the most common file system utilized by the Linux distribution establishing it as the default File system in the various distributions. Features such as allocation; which controlled ways through which storage blocks were allocated before disk write process begun to improve read and write performance, persistent- pre-allocation; using the function *fallocate ()*, to guarantee allocation of space and attempts to find contiguous space without writing to the disk to improve the performance of both writes and reads of streaming data and databases, checksums with multiblock allocation; which did not exist in Ext3, allowing Ext4 file systems to realize invalid entries on initial mount after system crash to avoid partial and out of order rollback of file systems, delayed allocation to reduce fragmentation while improving performance and improved timestamps in nanoseconds as compared to Ext3 which handled events in seconds timestamps (Salter, 2018). Ext4 manages content and file storage as a linear collection of blocks called block groups, by assigning storage in individual units called Blocks. A block consists on a collection of sectors of the disk with varying sizes ranging from 1KiB and 64KiB. These blocks are further grouped together into relatively large units to form Block Groups with an increased size of about 4KiB which is determined at mkfs time.

As an Advancement to the ext3, ext4 attempts to reduce performance loss and delay by keeping files of related content or components to that are in individual blocks into the same group. Thus, Files blocks that are scattered in storage is drastically reduced in this

file system. This is done to reduce the seek times. The file grouping of blocks is controlled by the Block Allocator. Thus, with the default size of block groups being 4KiB, each group will contain cumulatively 32768 file blocks, for a length of 128MiB block group. Pomeranz (2010) emphasized on establishing a relation between the block groups and size of the storage device in order to determine the number of block groups located on that device, as the dividend of the size of the device and the block group size (Pomeranz, 2010). Table 2.1 below provides the layout of a standard block group.



sandbox mode.

Table 2.1: Standard block group layout of ext4

Group 0 Padding	Ext4 Block	Supper Descriptors	Reserved Blocks	GDT	Data Block Bitmap	Inode Bitmap	Inode Table	Data Blocks
1024	1 block	Multiple blocks	Multiple blocks		1 block	1 block	Multiple blocks	Many more blocks

Being a Journaling File system, the ext4 has a journal that keeps record of updates in the metadata of the file system before updates are carried out in the file system. Serving as a redundancy in the case of OS crash, the Operating system accesses the journal and reads the content in order to either reprocesses the changes administered to the file system during and or before the Crash or roll back all transactions in the journal. For instance, as the table depicts the layout, metadata structures such as Directory entry, keep record of file names and inodes that store file metadata. The journal does not only record updates made to blocks such as the values being changed but holds records of the full block that is updated. When a new file is created, resulting in the update of the File system blocks, the journal must contain the updated record of the blocks which contain the Directory entry and inode data. With this journaling procedure, recovery of files even after formatting is possible; because on deleting a file, the file system only deletes the journal information, leaving the deleted content intact but not visible. Despite the aim of ensuring redundancy and easy recovery, this technique also provides hackers the opportunity to also recover deleted files. It is worth indicating that Ext4 was specifically design with backward-compatibility with Ext3 allowing Ext3 file systems to be easily upgraded and permit Ext4 drivers to automatically mount Ext3 file systems in a unique



# 20 KNUST



### **2.1.2 Ext4 Security**

The inefficient implementation of ext4 file system utilities is the main cause of metadata deletion failure. Earlier versions of the ext4 File system utilities wiped the partition automatically when a format operation is initiated, by default. But in actual implementation, it only wiped the index of the selected files or partition, leaving the metadata untouched. Subsequent versions of ext4 introduced an explicit wipe option, merged into the Android Open Source Project. A patch released in 2011 included a secure deletion module which raised a number of concerns to the integrity of the overall file system, as the ext4 file system separates metadata information from the blocks storing the target data. The secure deletion module would delete the data block permanently but fail to map the deleted data to its associated metadata which resulted in frequent file system integrity mismatch drastically reducing storage integrity as the file system attempts to reconstruct the files with persistent metadata due to the fact that the file system understood the missing block with existing metadata as a file system corruption (Corbet, 2011).

This improvement includes a `-w` option to strictly inform the `_ext4fs` utility to completely wipe the selected partition before initiating a format operation. Originally, the intention was to use the `"BLKSECDISCARD ioctl"` option to discard sectors of deleted data matching the target deleted partition in order to avoid leaving any traces of supposedly "deleted" data. Thus, if the explicit `-w` option is ignored then the formatting process will not erase

the target data on the selected partition, but that data will not be directly available to the user (Shu *et al.*, 2017).

## **2.2 Android Operating Systems Overview**

The Android OS is a mobile device operating system developed through modification of the linux Operating system. Thus, the Open Handset Alliance (OHA) designed the linux kernel 2.6 as the building blocks for the development of the Android Operating system. Originally developed by Android Inc. with the strategic mission of the company to penetrate into the mobile device space. Google purchased and took over further development of the operating system. With the aim of creating an openly available collaborative community, google made majority of the Android operating system codes available under the open source license, due to this, more vendors (typically hardware manufacturers) added and are still adding proprietary extensions to Android and customize it to differentiate their product from others. The main advantage to adapting Android is that, Android offers a unified approach to application development. This means that developers need only to develop for android in general and their applications will run on the numerous devices powered by Android.

The Android system has a default first physical hard drive `/dev/hdO`. By providing a Memory Technology Device (MTD) interface to act as the operating interface for the Flash device and Kernel for linux. (Rao and Chakravarthy, 2016). The Dalvik virtual machine use

the concept of "sandboxing" to enable multiple application interfaces to run independent of each other with unique process identification; each process is run in its own separate Virtual Machine. The applications are not allowed to interact each other unless given special permissions. Android application is a package in apk format with manifest file (describes essential information about your app to the Android build tools, resource file and Dalvik executables (dex). Within the device memory, core libraries, system and configuration files are contained. Within the internal memory data relating to application and user data is contained. (Rao and Chakravarthy, 2016). Execution of new applications requires a new process to be created and the applications do not interact. As a security feature, versions of the Android systems are developed to ensure that a unique process of data imaging performed on one type of android device cannot be repeated on subsequent devices.

### ***2.2.1 Android Operating System Architecture***

Android Operating systems is divided into four layers namely applications Framework, Libraries and Android run-time environment and Linux Kernel. With careful integration of each layer for optimal application development and execution. The Linux kernel which serves as the platform on which the Android Operating System was built consists of multiple hardware components device drivers within the android device. The libraries handle the main features of Android O.S; for instance, the database support for

applications to utilize within the android O.S is handled by the SQLite library, Web browsing functionality and credential processing is handled by the WebKit library, etc.

Android Run-time and libraries reside in the same layer within the Android operating system architecture; serving as a collection of core libraries for android developers to write applications using JAVA. The Android run-time also includes the Dalvik virtual machine which performs sand boxing for each application process within the Android operating system. Android rune-time is optimized for Android battery-powered devices with memory and CPU power limitations, allowing it to adapt based on availability of resources. Application Framework provides a wide range of capabilities of the Android O.S to application developers. Application, is the top layer component that contains built-in android applications as well as components for installing other applications; this layer contains all the information pertaining to application installation and usage such as contacts browser, phone etc. Figure 2.1 illustrates the Android operating system architecture. Which is illustrated in five views; The application or User Interface, the application Framework/ Developer view, Libraries, Android run-time and Linux kernel, showing the relationships between various levels of the android Operating system architecture.

Applications refer to the programs installed to allow interaction between the user and the device through Graphical User Interfaces (GUI). Applications are written to be installed at this level only. Application framework consists of the structure/components associated

with the applications displayed to the user; this defines regulations and management controls. They also provide a large range of high-level services to applications allowing developers to write instructions for applications to access these services. Android Runtime is the Dalvik Virtual machine designed and optimized for android. Libraries consist of all core libraries that facilitate android device usage and support for external features and services. The Linux kernel is associated with the device. Specifically, the bottom layer is Linux, providing a level of abstraction between the hardware and all essential drivers.

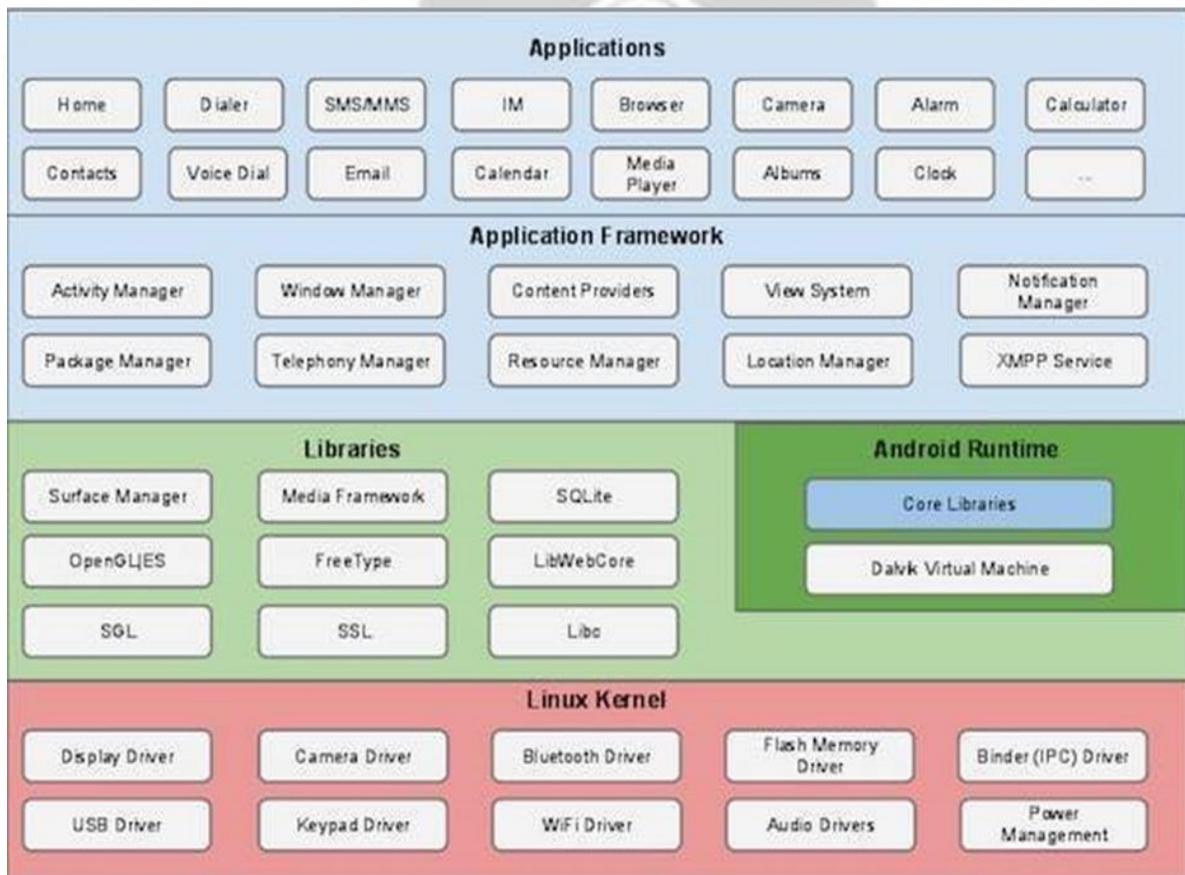


Figure 2.2: Android O.S. Architecture (Source: [www.tutorialspoint.com/android](http://www.tutorialspoint.com/android))

Android file systems unlike windows, lacks the concepts of drives. Built on the Linux environment (Dalvik Virtual MACHINE), its hierarchy originates from the root (/). Thus, the entire content of the Android file system is built on a single hierarchy which is the root. With the development of different versions of Android, stems the large variations in file system structure of each build version; but the default file system implemented within many android mobile devices is the JFFS2 (Journal Flash File System version 2) replacing the original JFFS.

With many of these variations, the entire tree structure is hidden from the default user of the mobile device. Requiring that the user gains root access in order to view and manage every component of the file system. Android files system for mobile devices, tablets and other devices consists of six main partitions, with each having unique functionality contributing to the smooth operation of the device. /boot, ./system, ./recovery, ./data, ./cache, ./misc with the external SD cards having /sdcard and ./sd-ext. Each partition divide into different sizes depending on their content. Using the ADB commands for Android, each partition and their corresponding sizes can be identified.

The /boot partition of the device includes the android kernel and ramdisk; without this partition, the device will not boot. /system partition contains the entire android operating system except the kernel and ramdisk, it also includes the android Graphical user Interface and all default system applications that come pre-installed on the device. Wiping this

partition will also render the device unbootable. /recovery partition is designed for backup functionalities; making it an alternative boot partition for the system to enable the device boot into recovery as well as advance maintenance operations. /data refers to the userdata partition which includes the entire mobile user content, performing a reset will request the complete wipe of this partition. /cache which is designed to keep record of user browser sessions, frequently accessed data and application contents. /misc which contains miscellaneous device settings in the form of switches such as USB configurations and other hardware settings. /sd card which is not an internal device memory, keeping record of another user content

Amongst the partitions, this project focuses on the properties and content of the /data and /SD card which contain user personal identifiable information, viable for any form of identity theft and criminal act once its content is recovered as well as storage and deletion techniques implemented at this layer for user information security, thus we further our search into the details of Android storage and file system structure for data records.

### ***2.2.2 Android File Systems***

The Virtual File System (VFS) servers as an abstract layer allowing all file, directory and system operations to flow through the Linux kernel in android; containing a large number of file systems, each serving as an implementation of the VFS. There is a separate kernel module assigned to each file system and registers operations supported by the VFS. With

the concept of high-level abstraction, separated from the actual implementation, including a new file system simply involves writing another kernel module to fit within a unique and independent tree structure. Each kernel module is either part of the kernel by default, or dynamically loaded on demand. Thus, the android kernel by default contains a subset of varied file systems ranging from the Journal File System (JFS) for AIX (IBM Unix flavor) to Amiga File System. Within each file system, directory path and internal functionalities are hidden from the user, since the kernel is responsible for, and performs all the operations when a file system is mounted. (Kim *et al.*, 2012)

The file system modules that are compiled is determined by the kernel configuration file as well as whether they are built into or dynamically loaded into the kernel. Resulting In the presence of only the file system modules that are relevant to its operations in the Android Kernel. Therefore, the entire Linux File system is not integrated into their tree structure, a common process seen in many of the Linux distributions, as some of the file systems are dependent on the hardware architecture. The file system supported vary when it comes to android devices but the commonly found flash memory file systems are:

exFAT - The extended File Allocation Table is a proprietary file system targeting flash memory by Microsoft. It does not form part of the standard Linux kernel due to licensing requirements. Even so, android support for such file systems is included by some

manufacturers.

F2FS - Introduced by Samsung and referred to as Flash-Friendly File system in 2012. It is an open source Linux file system.

JFFS2 - The Journal Flash File System version 2 serves as the AOSP (Android Open Source Project) kernels' default flash file system since Ice Cream Sandwich. Serving as a replacement for the JFFS.

YAFFS2 - Yet Another Flash File System version 2 served as the default AOSP flash file system in kernel version version 2.6.32. It no longer has support in new kernel versions and will not be identified in the source tree for the recent versions of the kernel from kernel.org. Support for this particular file system solely depends on the device vendor.

Aside the above flash memory file systems, Android devices support media -based files systems such as:

EXT\* - Where \* represents numbers 2,3 or 4. Serving as the standard Linux File system, the EXT4 is the most recent version. Dated back to 2010, EXT4 often served as a replacement for YAFFS2 or JFFS as the internal flash memory file system for android devices.

MSDOS – The Drivers for MSDOS support FAT12, FAT16 and FAT32 file systems. VFAT – Serving as an extension to FAT12, FAT16, and FAT32 file systems, VFAT kernel module is always identified in pair with MSDOS module. External SD Cards are mostly formatted

using Virtual FAT.

The above are media-based file systems but VFS supports pseudo file systems; they are not media based. Some pseudo file systems are important to android and thus are given support on the Linux kernel for android devices.

cgroup - The cgroup or control group is a pseudo file system that provides a way to access, define and alter various kernel parameters. Although cgroup is a pseudo file system, there are a number of varied control process groups. Thus, if your android devices have support for process control groups, you will be able to find the group list in the file `/proc/cgroups`. cgroups are used by android in user accounting (acct) and CPU controls (cpuctl).

rootfs - Serving as root file system ("/") mount points.

procfs - Reflecting a variety of kernel data structures, procfs file system can be identified in the `/proc` directory; using various operations to read live kernel data. Process IDs (to be specific, the thread group head process ID) are reflected with the number directories according to the task running. The file `/proc.filesystems` will generate a collection of registered file systems Any file system appended with `NODEV` are pseudo file systems, therefore have no related device. With kernel parameters of which most are tunable is contained in `/proc/sys` (Kim *et al.*, 2012).

sysfs – All devices known by the kernel is reflected using an object-oriented structure for the device model using sysfs file system. The sysfs is usually mounted on the /sys directory, which is normally mounted on the /sys directory. Upon discovery of a new device, an object is built in /sys/devices directory, using a network socket to establish a communication channel with the new device to the udevd daemon, which will build an entry in /dev directory. Media-based file systems and their kernel objects are contained in the /sys/fs directory. Each kernel module loaded is contained in the /sys/module.

tmpfs – This is also a pseudo file system mounted in /dev, and because of the nature of pseudo file systems, any data in /dev directory will be lost when rebooted

While we have listed number of file systems, there are still a large number of file systems available beyond this scope. Also, without privilege/root access, these file systems are hidden from plain view. For some devices such as Samsung Galaxy SIII, and option called USB debugging is available under developer options. Enabling this feature will allow a user to establish device connections using the adb (Android Debug Bridge ). Even with this feature and process followed, it will only provide user level privileges, limiting access to directories and some linux commands.

### ***2.2.3 Hierarchy of the Android File System***

Users of Linux are exposed to another aspect of the file hierarchy for linux. Usually represented as a single tree, with the root (/) as the beginning and below are a large

collection of directories containing files and individual files outside directories. Unlike the windows operating system, the “logical drives” concept is not applied in linux.

Alternatively, file systems are mounted within a directory to form a single tree. The file system in media-oriented file systems are represented as partitions, irrespective of whether they exist physically on the local device or remotely. Every component is integrated within that singular hierarchy that is initiated by the root. The “path” simply represents the cluster of directories within which a file/target is located. With “/” appended to each file path, resulting in an absolute file path. Thus, a path /mnt/sdcard simply results in the sdcard directory which can be found within the mnt directory. Names and directory properties may vary, but the rule remains the same. One major rule to remember is; each directory represents a file system mount point. Android only makes visible the file systems which are accessible to the current user privileges, these file systems are part of the file hierarchy. (Hoog, 2011).

Thus in order to view the entire hierarchy, you will require root access/administrator access; a unique user account that allows system level changes. When a device is “rooted”, it means accessing the administrator or root account. This access is not automatically given in order to avoid security compromise through oblivious errors. But for users with some level of system level knowledge and linux system administration, the underlining structure of Android operating system is open and accessible. The following list shows

the top-level android directory structure as released in AOSP Jelly Bean.

**acct** –Mount point for user accounting features using cgroup

**cache** -/dev/block/mtdblock2 (may vary in other versions) mount point. Thus, there is a

relationship between this partition and the cache size. **d** -/sys/kernel/debug symbolic

relationship.

**data** -/dev/block/mtdblock1 mount position. The mount point for the

/dev/block/mtdblock1 partition. **default.prop** –

Contains a list of default properties

**dev** – Contains a list of all applications available to the device. Serving as a the tmpfs

mount position. **etc** -/system/etc symbolic relationship.

**init** -Responsible for processing the init.rc file which will access other similar files. The init

is executed by the kernel at the end of the android booting process. This is important

because it is the source of information concerning the summary of configuration of the

booted Android device.

**mnt** -Aside serving as mounts for external and internal storage (SD cards), this directory

is responsible for mounting other unique file systems. Besides the mounts for the internal

and external SD cards, these directory serves as a mount point for other file systems such

as the /mnt/asec, a directory which is part of the Android Security.

**proc** -Serves as a mounting point for the procs, providing data structures for the kernel.

Also serves as a source of information for programs such as ps,lsof and vmstat.

root – The source or initial directory for the root account. **sbin** -Contains

binaries for a variety of system daemons. **sdcard** - /mt/sdcard symbolic link.

**sys** – Reflects the object structure for the device kernel, high level of abstractions within the files of this directory and requires adequate knowledge of the kernel device.

**system** -Serving as /dev/block/mtdblock0 mount position. Contains default directories seen in a standard linux distribution root directory such as xbin,lin,etc an usr.

vendor – Symbolic relationship for /system/vendor

### **2.2.4 The Android Storage System**

Personal computers utilize the magnetic disc storage technology as storage devices for their system. This is not the same for android devices, which store data on the SoC containing an embedded Multimedia Controller(eMMC) as well as the Solid -State Drive (SSD) as internal memory storage (Altuwajri and Ghouzali, 2018). This type of memory medium is a flash memory, specifically, a NAND type flash memory, which does not provide a random-access external address bus to its I/O interfaces. Thus, ensuring that data is read block wise, with each block containing many bits. This type of medium

contains strictly deletion procedures that ensures that even if a part of a block is to be deleted, the entire block must be overwritten. In order to prevent data that is not specified for deletion to be lost, it does not erase the content immediately, but labeled "unallocated" by the system. Resulting in data remaining after deletion. Total data erasure is achieved only if the entire block is overwritten.

Flash memory has a lifecycle specified by a finite number of program-erase cycles, with many commercially manufactured flash media designed to withstand about 100000 program-erase cycles before integrity and performance declines gradually. These Flash media will therefore retain deleted data until it is overwritten by another operation of possibly equal size, resulting in a longer residence time or data after deletion (Reardon, Basin and Capkun, 2013). Residence time is dependent on the size of the memory and frequency of usage; the larger the size of the memory, the longer the residence time since it requires that the entire memory is occupied before data replacement can occur, and dependent of how frequently the medium is use for storage and deletion.

Unlike traditional hard disks, data-writing management is an important characteristic of Flash memories, which prevents the device from performing a direct overwrite (in-place overwrite operation). Thus, if the mobile device file system attempts to overwrite an existing block of data on the flash memory, its on-board controller will redirect the write operation to a new block while flagging the old data block as "unallocated". Resulting in data retention flaws transparent to advanced android and file system enthusiasts (Shu *et*

al., 2017).

### 2.3 Digital Forensics and Related Works

The adjective *forensic* comes from the Latin word *forensis*, meaning “in open court” or “public.” Describing something as forensic usually meant that it has to do with finding evidence to solve a crime. It could also mean that it has to do with the courts or legal system. Scientifically, Forensic is the process of using scientific knowledge for collecting, analyzing, and presenting evidence to the court. Several definitions of Digital Forensic have been given by different renowned scientist. Notable among them are as follows

“Digital forensics is a specific, predefined and accepted process applied to digitally stored” (Palmer, 2002)

Kohn predefined digital forensic as “accepted process applied to digitally stored data or digital media using scientific proven and derived methods, based on a solid legal foundation, to extract after-the-fact digital evidence with the goal of deriving the set of events or actions indicating a possible root cause, where reconstruction of possible events can be used to validate the scientifically derived conclusions”.

Reith and colleagues define computer forensics as “the collection of techniques and tools used to find evidence in a computer” (Reith, Carr and Gunsch, 2002). The same authors, however, explain digital forensics as a broader concept to include (p. 2): Implementation of proven and scientifically derived methods aimed at the collection preservation

authentication/validation, interpretation/analysis and documentation as well as presentation of digitally obtained evidence aimed at the facilitation of reconstructing of events identified to be of criminal intent or aiding to anticipate unapproved actions targeted at disrupting planned operations. (Reith, Carr and Gunsch, 2002).



Wolfe defined computer forensics as *"A methodical series of techniques and procedures for gathering evidence, from computing equipment and various storage devices, digital media, that can be presented in a court of law in a coherent and meaningful format"* Raghavan (2013, P. 91) described digital forensics as "a branch of science that involves the application of scientific principles to the investigation of artifacts present in one or more digital devices in order to understand and reconstruct the sequence of events that must have transpired in generating the said artifacts"(Raghavan, 2013). "Within the past few years, a new class of crime scenes has become more prevalent, that is, crimes committed in the domain of electronic or digital devices especially the cyberspace domain. Globally, agencies dealing with criminal justice are faced with an increased need to investigate crimes perpetrated partially or entirely over the internet or other electronic media. To effectively identify, while preserving the sanctity of electronic evidence, specific procedures are adhered to with the required resources readily available. The evidence ranges from images of child pornography to data that is encrypted purposed for various criminal activities." (Lee, Palmbach and Miller, 2001)

Digital forensics emerged in response to the drastic increase in crimes committed under the scope of anonymity. Perpetrators use computer systems as a medium of committing crime(means) or as the object or target of a crime. In some cases, the computing device serves as a storage or repository for the evidence relating to the crime committed.

Procedures and findings on digital forensics date back to the early 1984 when FBI and numerous law enforcement agencies began to design and implement software to analyze computer evidence. Research community in the likes of the Computer Analysis and Response team (CART), the Scientific Working Group on Digital Evidence (SWGDE), the Technical Working Group on Digital Evidence (TWGDE), and the National Institute of Justice (NIJ) have since been formed with the aim of discussing computer forensics science as a discipline with the focus of standardizing the approach to investigations and examination. (Noblett, Pollitt and Presley, 2000).

It is obvious from the definitions above that, digital forensics employs uniform and well – defined step – wise approach for identifying and retrieving of evidence, preserving, analyzing and interpreting the evidence in an uncompromising manner with the intent of reconstructing the events for a possible admissibility in the court of law. As stated by Reith and colleagues, digital forensics unlike computer forensics encompasses several digital sources and in lieu of this research thesis, mobile forensics and its related works are considered (Reith, Carr and Gunsch, 2002).

### ***2.3.1 History of Mobile Forensics***

Mobile forensics is a branch within digital forensics focusing on access of digital evidence or data from mobile devices under sound conditions and ensuring the reproducibility of the results obtained. Mobile device usually refers to mobile phones; however, can also

represent any digital device that has memory and communication capability such as GPS devices, tablet computers and PDA devices. Due to its wide usage, criminals are more focused on the acquisition of Personal Identifiable Information from mobile devices since mobile devices are now used to communicate and store corporate and personal information as well as for performing online transactions

The analysis of mobile devices using forensic means originated in the early 2000s. Law enforcement of many countries began to comprehend the direct and indirect involvement of mobile devices in crimes. With the continuous increase in production and access to mobile devices as well as the wider array of communication platforms that these devices can support, the need for forensic examinations grew (Casey, 2011).

Law enforcement agency utilizes mobile services whilst criminals take advantage of the wide capabilities of mobile devices to perpetrate crimes (Ahmed and Rizwan, 2004). Both internal and external storage capacities have increase due to the demands for more powerful minicomputer type devices (Tsukayama et al., 2012). Initial attempts to analyze the content and activities of mobile phones implemented a similar technique to that of computer forensics investigations; involving analysis of these devices using a screen and photographing important data. But the constraints on this approach was its time consumption, the increase in storage capacity of mobile devices and finally the increase in the number of devices analyzed required that the data capturing component had to

have a continuous upgrade of storage capacity. Thus, investigators requested for a more efficient means of data capture.

Many professional mobile forensics examiners often used cell phone backup and synchronization software to capture device data on to a forensic computer by imaging and in some cases performing computer forensics on a hard disk of a suspect's computer where device data had been synchronized. However, this process resulted in a number of challenges; the software could make entries to the phone as well as reading it, and unable to recover deleted data (Curran *et al.*, 2010). Therefore, mobile device forensics provide some inherent challenges in relation to evidence access and technical setbacks. For instance, analyzing cell sites which defines the use of mobile phone and coverage is not an exact science. According to Murphy and Cynthia (2013), by estimation it is possible to determine cell site zone where a call may originate but still yet to discover with a degree of certainty, that mobile phone call was made or received from a unique location e.g. a call intercepted from a resident address (Murphy and Cynthia, 2013). To maximize utility, original equipment manufacturers (O.E.M) improve or in some cases alter mobile devices form factor (Hardware dimensions and specifications), as well as many internal logical operations.

Experts in forensics discovered ways of recovering deleted data by implementing a "flasher" or "twister" boxes, tools developed by Original Equipment Manufacturers to

“enable administrative/ elevated mode access to” a mobile device’s memory for updates and debugging. However, this process also came with its own set of challenges; flasher boxes are invasive and can alter data; as well as complex process involved in Flashing.

Since the tools were not developed based on forensics, do not provide a means of ensuring data integrity through hash verifications nor (in most cases) audit trails (Thackray, 2012). Thus, for forensic analysis of physical devices, better alternatives were required.

To eliminate the constraints of the stated procedures, commercial tools were developed to allow examiners to acquire a bitwise (bit to bit) of the phone's internal and external memory with minimal alterations and analyses to each image separately (Casey, 2011).

With the passage of time, these commercial techniques have been improved, including techniques for deleted data recovery from user’s mobile device, made possible using special tools. Furthermore, these commercial tools have even provided automated means of extracting device contents; making it possible for individuals with very little training to perform basic extraction for data review purposes.

### **2.3.2 Android Mobile Forensics**

Android Operating Systems (OS) gained an increase in popularity in recent times. Its function is closely related to the basic functionalities for which the desktop/laptop computers were developed. Today, there are billions of Android users all over the world (Rao and Chakravarthy, 2016), revealing its high demand by people because of the flexibility of service it provides; Users can develop and easily personalize documents, surf the internet, read and send emails, contact people via phone calls, SMS and MMS, personal scheduling as well as keeping personal information. The global Acceptance of mobile devices have made them intrusive in our daily activities, providing forensic examiners huge amount of information from the devices they study because these android devices and applications provide flexibility and ease of use for communication, web browsing, sharing and social interactions amongst friends, family and various groups personally or professionally. Also, individuals might assume that their contents are secure and inaccessible.

On the other side, the 'addiction' to these mobile devices make them vulnerable places for cyber criminals to gain Personal Identifiable information (PII). In order to identify a sequence of activities carried out by a suspected person, forensic examination of mobile devices is mandatory for any cybercrime examinations. Many researches have cited the growing importance of mobile device forensics as well as ways of obtaining vital information through and from mobile devices for different purposes (Roy, Khanna and

Aneja, 2017). The Authors of this report presents forensic analysis of android mobile systems with respect to all kinds of memory areas of the phone as well as contact information, call history and exchanged information via social platform applications. With the vast availability of popular and less popular forensic tool, the report aims to bring out a detailed methodology of performing forensic analysis in a sound and secure way to produce results that are reproducible and viable for any legal process. This report could serve as a starting point for several android users, forensic analyst and investigators by providing a detailed analysis and understanding of the Android File System architecture and the processes involved in Identifying, Acquiring and analyzing memory and system information of Android devices in a forensically sound manner.

## **2.4 Digital Forensics Models**

In a digital forensic investigation, the sanctity and integrity of the evidence herein referred to as digital evidence is critical to any forensic investigation, thus priority is given to the procedure and means of evidence. This concept is buttressed by (Yusoff, Ismail and Hassan, 2011) that, the adopted process of computer forensic investigation has direct impact on the results. Selecting the wrong investigative process may result in inconclusive results, therefore, resulting in invalid conclusion. Evidences obtained in an unstructured process may result in the risk of not being admissible in the court of law.

This paper begins with existing digital forensics investigative models, analysis of existing model to identify the strength and some weakness inherent in those investigative models.

#### **2.4.1 Abstract Digital Forensics Model (ADFM, 2012)**

Reith and colleagues proposed the Abstract Digital Forensics Model (ADFM) (Gary, 2001) which was both an inspiration from and enhancement of the Digital Forensic Research Workshop (DFRW) model which had the following phases (Identification, Preservation, Collection, Examination, Analysis and Presentation) (Reith, Carr and Gunsch, 2002). The ADFM model introduced three new phases (Preparation, Approach Strategy and Returning Evidence) to the six (6) phase model proposed by the DFRW model making the ADFM a model with nine (9) phases which are; Identification, Preparation, Approach Strategy, Preservation, Collection, Examination, Analysis, Presentation and Returning Evidence.

The **Identification** phase is the initial step where different indicators are used to determine the incident type. **Preparation** phase which is the next stage deals with preparing tools, procedures, search warrants, as well as monitoring authorizations and management required to legalize further investigations. **Approach Strategy** phase follows next in the sequence aiming to maximize collection of untainted evidence at the same time minimizing impact to the victim. Next is the **Preservation** phase which isolate, secure and preserve the state of physical and digital evidence.

**Collection** phase, the physical scene and duplicate digital evidence is recorded using standardized and accepted procedures. In – depth systematic search of evidence relating to the suspected crime is performed at the **Examination** phase to construct detailed documentation for evaluation. Following the Examination phase is the **Analysis** phase

which determine significance, reconstruct fragments of data and draw conclusions based on evidence found and also to support the hypothesis of the crime. **Presentation** phase, results and conclusions are compiled to provide explanation of conclusions which is mostly done in such a way that a layperson can comprehend and finally the **Returning Evidence** phase which ensure physical and digital property is returned to proper owner and determining what criminal evidence must be removed.

### **Advantages**

1. It is a diverse methodology which can be applied to a range of digital devices from calculators to desktop computers, or even some future digital devices.
2. It is a generalized methodology that can assist non-technical observers to identify with technology.
3. Potential for incorporating non-digital, electronic technologies within the abstraction

### **Disadvantages**

1. The model defines categories which may be too general for practical use.
2. There is no easy or obvious methodology for testing the model

#### **2.4.2 The Enhanced Digital Investigation Process Model (EDIP)**

The Enhanced Digital Investigation Process Model mostly referred to as EDIP was developed by Venansius Baryamureeba and Florence Tushabe. The EDIP is an investigative model based on a previous model by Brian Carrier and Eugene Spafford known as the Integrated Digital Investigation Model (IDIP). EDIP seeks to redefine the forensic process and its progression by expanding the deployment phase in the IDIP model. The expansion as accounted by Baryamureeba and Florence (2004) includes the physical and digital crime investigations while introducing a new phase dedicated to tracing back to the computer (the primary crime scene) that was used as a tool to commit the offense. The EDIP model has two categories of crime scenes which is the suspect's or the primary crime scene and the victim's or the secondary crime scene (Baryamureeba and Florence, 2004).

This model has five major phases namely, readiness, deployment, trace back, dynamite and review. The model starts with the **readiness** phase where operations and infrastructure readiness are undertaken. Thus, the needed human capacity is properly trained and equipped to deal with the situation and a check is done to ascertain that the underlying infrastructure is enough to deal with the incident when it occurs. The **deployment** phase provides mechanism for the detection and confirmation of an incident. This phase has five sub – phases which includes detection and notification phase where incident is detected and the respective individuals or authorities notified appropriately, Physical crime scene investigation where physical examination of the scene

is performed to unearth potential digital evidence, a digital crime scene investigation is done through electronic examination of the scene to acquire digital evidence and possibly an estimation of the extent of the impact or damage is done. The confirmation sub phase is when the incident is confirmed, and authorization given to obtain legal approval to carry out search warrant and further investigations at suspect's premises. The submission sub phase involves presenting the physical and digital evidence to legal entities or corporate management. The **traceback** phase tracks down the operations of the suspect's physical crime scene resulting in the identification of the devices that were used to perform the act. This phase encompasses two sub – phases; digital crime scene investigation and authorization phase. Succeeding the traceback phase is the **Dynamite** phase which conducts investigation at the primary crime scene with the aim of collecting and analyzing items that were discovered at the scene to obtain further evidence that the crime originated from to help apprehend potential culprits. The entire investigative process is reviewed, and possible areas of improvement is identified in the **Review** phase (Baryamureeba and Florence, 2004)

Though Enhanced Digital Investigative Process model is an improvement over the IDIP but there exists some ambiguity with some activities carried out in some phases or some clarification have to be provided. There is a submission sub – phase under the deployment phase which states that; presentation of the physical and digital evidence is made to the legal entity or corporate management (Gary, 2001). The unanswered question here is, what are they to do with that

presented material and how does the outcome of that sub – phase affects the investigative process?

There seems to be duplication of activities. E.g. Digital crime scene investigation activity appears under the Deployment phase, Traceback phase, as well as Dynamite phase. It may be argued that, due to the iterative nature of the investigative model proposed. According to Gary (2001) EDIP separates the investigations at the primary and secondary crime scene while depicting the phases as iterative instead of linear (Gary, 2001). That notwithstanding, it would have been appropriate to give the output at each phase or per each activity in the phase so as to draw some distinction and also serve as checks for the investigation.

#### **2.4.3 Computer Forensics Field Triage Process Model (CFFTPM)**

Rogers *et al.*, (2006) proposed this model. It is an onsite or field approach for providing the identification, analysis and interpretation of digital evidence in a short time frame, without requirement of having to take the system(s) media back to the lab for an in-depth examination or acquiring a complete forensic image(s). The model is needful in situations where quick information and investigation leads outweigh the need for an in – depth analysis of all the potential digital evidence back in a laboratory. Hence the model is mostly used on the field or at scene (Rogers *et al.*, 2006).

The three basic components of forensic investigation by Kruse and Jay (2002) also called the “3As” of computer forensics investigation guided the formulation of CFFTPM (Kruse and Jay, 2002). The CFFTPM was developed with the following foci; find useable evidence immediately, identify victims at acute risk, guide the ongoing investigation

identify potential charges and accurately assess the offender's danger to society while at the same instance protecting the integrity of the evidence and / or potential evidence for further examination and analysis (Reith, Carr and Gunsch, 2002). The CFFTPM has six (6) main phases with two (2) of the phases having three (3) sub – phases each and these are, Planning, Triage, User Usage Profile (Home, File Properties, Registry), Chronology Timeline, Internet (Browser, Email, IM) and Case Specific.

This model begins with the **planning** phase where proper prior planning is undertaken with the investigator formulating some indicators and directions highly probable to result in successful investigation. After the Planning phase is the **Triage** phase where prioritybased activities are executed. Hence items, pieces of evidence or potential containers of evidence that are highly important or the most transient are first dealt with. In the **User Usage Profile** phase, actual examination and analysis is performed on evidence found on digital media in order to link the evidence to a specific, identifiable suspect. **Chronology Timeline** phase builds the crime incident from organized chronology to sequence the probable crime activities mostly by using some timing model such as the MAC (Modification, Access and Creation) times. With the **Internet** phase, examination of artifacts related to internet activity such as Instant Messaging (IM), e – mail and web browsing is performed. The final phase is the **Case Specific Evidence** phase

whose success largely depends on the skill of the investigator employing the model in an investigation. In this phase, adjustments are made concerning the focus of every

examination of that case as well as reconciliation of conflicting requirements are done in a manner to suit each specific set of circumstances.

### **Advantages**

1. This model is much concerned about time, hence help to undertake quick information and investigation in a time critical situation.
2. This model is used to conduct investigation on scene which provides additional benefit of having feedback loop with the investigator(s).
3. It also affords computer forensics analyst to modify their searches right on the scene based on input from the primary investigator(s) as well as those in direct contact with the suspect.

### **Disadvantages**

1. This model is only appropriate for investigation conducted at the scene
2. It may be an incomplete investigative model should the case under investigation require additional work to be done off the scene.
3. This is a time critical model, hence in the usage of this model some investigative process or phase may be compromised thereby affecting the results or outcome.

#### **2.4.4 Generic Computer Forensics Investigation Process Model (GCFIPM)**

Yusoff came up with a digital forensic investigation model known as the Generic Computer Forensics Investigation Process Model after reviewing existing forensics investigation models from 1984 to 2010 (Yusoff, Ismail and Hassan, 2011)

The models which were reviewed as well as the phases under those models were giving unique ids without being oblivious of the relationship between the models and their respective phases. After assigning the ids, the tasks and not just the nomenclature which are performed under each phase was considered resulting in a five (5) generic grouped phase proposed model. The GCFIPM phases include, Pre – process, Acquisition & Preservation, Analysis, Presentation and Post – process.

The Pre – process phase deals with activities that are carried out prior to the actual investigation and official collection of data such as getting the necessary approval from relevant authority, etc. Under the Acquisition & Preservation phase, tasks related to identifying, collecting, transporting, storing and preservation are performed. Next is the Analysis phase which is considered as the core of the forensic investigation processes and various types of analysis are performed on the acquired data to identify the crime source and possibly the perpetrator of the crime. The Presentation phase is where various outcomes of the Analysis phase are documented and presented to authority in a format which is easily understood and mostly backed by sufficient and acceptable evidence.

Finally, is the Post – Process phase where proper closing of the investigation exercise is done, rightfully owners are given the needed digital and physical evidence and review of the investigation process is done for lessons to learnt and improvement be done for future investigations.

### **Advantages**

This model puts phases of several models into groups making the model suitable or applicable to diverse types of forensics investigations. The model serves as a broad or generic framework which can provide a good starting point for the development of new digital forensics investigation model.

### **Disadvantages**

The phases within this model was formed by grouping phases of other models which eventually introduces duplicate activities in the grouped phases. Due to the generalized nature of this model, it is considered more of a guideline framework than a model.

## **2.5 Digital Forensic analysis for Android devices**

According to the IDC's report for the second quarter in 2015, Android devices has become an important data source in forensic investigation due to its popularity and a potential rich repository of user data. In these android devices however, the prime rich data

repository is the memory which maybe either internal or external. With respect to the above report, (Li, Xi and Wu, 2016) concluded that, most current studies consider only flashing the NAND card and not paying attention to eMMC card. Hence, their research paper sought to design an Android Forensic tool which considers both NAND and eMMC card. Data acquisition from an Android device can largely be divided into software-based method and hardware-based method. With software-based methods, some are for logical acquisition whilst others are for physical acquisition of data. This study however, uses physical acquisition method, designs a general forensic tool and focuses on the differences of acquisition between NAND and eMMC.

Li and colleagues conducted the physical acquisition using a flash recovery partition by booting the device into fastboot mode and using the command, "flash<partition> [filename]" to flash the specified recovery image into the recovery partition. The acquisition was done for devices with MMC partitions and those with MTD partition as well. Giving more attention to user data partitions. The image analysis was performed with emphasis on the user data partition image and the comparative analysis results tabulated in the paper (Li, Xi and Wu, 2016).

This study considers not only NAND card but all eMMC and performs collection and analysis thereof at the same time since according to the authors, prior works have not considered the MMC partition during physical acquisition though no additional facts were

elicited to buttress such claim. The authors concluded that a digital forensic tool was written in C++ but failed to provide details of the forensic tool with respect to its name,

its source (where it would be downloaded), whether on open source tool or a commercial one (Li, Xi and Wu, 2016).

## 2.6 Security Metrics for the Android Ecosystem

This paper provides a detailed work in suggesting the essential or crucial role the stakeholders or entities identified as players in the Android Ecosystem concerning the security of Android devices. The notion of an ecosystem therefore denote some responsibilities on the stakeholders in ensuring a secured Android Ecosystem, (Wagner *et al.*, 2015) proposed three metrics code named "FUM score" and are detailed below:

- i. F: measures the proportion of running devices free from critical vulnerabilities over time.
- ii. U: measures the proportion of devices that run the latest version of Android shipped to any device produced by that device manufacturer.
- iii. M: measures the mean number of outstanding vulnerabilities affecting devices not fixed on any device shipped by the device manufacturer.

In order to establish "FUM score" metrics, three key related issues namely; threat model, data and android ecosystem are considered. With respect to the threat model, the paper was concerned with vulnerabilities such as privilege escalation flaws and data privacy

intrusion which allow an attacker to remotely access the smartphone to gain significant permissions. Hence, a three attack vectors namely the installation attack vector, the dynamic code, loading attack vector and the injection attack vectors were perused.

With the issue related to Data, two sources of data were used to measure the security of Android:

- i. Information on the critical vulnerabilities found to affect particular versions of Android and
- ii. Information of the distribution of Android versions over time.

These two data items were obtained from two scenarios; researchers signing a legal agreement with the University of Cambridge to obtain data from contributors and contributors submitting a subscription validator to participate in the analysis performed by researchers. Thus the former data item made sure that case sensitive data was not made available and the latter data item allowed case sensitive data availability, according to Wagner and colleagues, + can then be combined to determine the proportion of devices at risk of attack from specific vulnerabilities (Wagner *et al.*, 2015).

For the Android Ecosystem, the concentration is on the creation and distributions of updates which fix vulnerabilities and the paper points out five entities or groups that forms the Android Ecosystem; transparency, purpose, withdrawal, consent and accountability which contributes towards Android updates; the network operators, the device

manufacturers, the hardware developers, Google and the upstream open source projects such as the Linux Kernel, Open SSL and BouncyCastle cryptography libraries.

(Shabtai *et al.*, 2010)

undertook a research which provides a security assessment of the Android framework by identifying high-risk threats to the framework whilst suggesting some-security solutions for mitigating them. In the paper, the Android security mechanism was considered by looking at Android OS architecture consisting of the Linux Kernel, Android native libraries, Android runtime, Application framework layer and the topmost application layer in a hierarchical order respectively. Motivated by the Android OS architecture, the authors looked at the several security mechanisms such as privacy enhancement techniques and compatibility with android OS vulnerability updates incorporated into the Android framework fundamental features and Android specific mechanisms.

Linux mechanism looked at the security at the Linux layer which prevents one application from disturbing another as well as preventing one application from accessing another's file. The environment features consider preventing Memory Management Unit (MMU) and preventing buffer overflows and stack smashing through Type safety as well as mobile carrier security features preventing phone call theft. The Android-specific mechanisms include the following security mechanism; application permissions, component encapsulation, signing applications and the Dalvik virtual machine.

For the Android Framework Security Assessment, the authors discovered that, a normal state Android device is well-guarded given that an attacker can replace neither the core components nor the kernel without manipulating the hardware. However, to alter operating system components is to identify a vulnerability in one of the kernel modules or core libraries. The Android threats were grouped into five (5) threat clusters:

1. Threat cluster 1 compromises availability, confidentiality or integrity by maliciously using the permissions granted to an installed application
2. Threat cluster 2 occurs when application exploits a vulnerability in the Linux Kernel or system libraries
3. Threat cluster 3 relates to application that usurp private or confidential content such as SD card, eavesdropping or wireless communication remotely
4. Threat cluster 4 involves attacks targeting draining a mobile device's resources such as disk storage, RAM, CPU, etc.
5. Threat cluster 5 finally deals with compromised internal or protected networks.

The publication concluded that, there are numerous security mechanisms incorporated in Android aimed to tackle a broad range of security threats. Security counter measures such as person identifiable information encryption and establishing secured communication channels for data exchange were proposed which could be implemented to harden

Android devices and enable them to cope with high-risk threats. The paper however did not specify how the implementation of those proposals could be done and the technical and economic feasibility of such implementation.

According to the paper, the three metrics  $f$ ,  $u$  and  $m$ , together measure the security of a platform with respect to known vulnerabilities and updates. In a nutshell, this paper is premised on the assumption that a secured Android device depends on the timely delivery of updates to fix critical vulnerabilities which the paper demonstrated that there is a bottleneck in the delivery of updates in the Android ecosystem by the manufacturers citing little incentive as the reason. In addressing the issue which will in the long run enhance the security, the FUM security metric was developed to quantify and rank the performance of device manufacturers and network operators, based on their provision of updates and exposure to critical vulnerabilities.

## **2.7 Emerging security threats for mobile platforms**

Delac and colleagues suggested that, there is a high level of similitude between modern mobile platforms and that of traditional OS for PCs resulting in some security-related challenges (Delac, Silic and Krolo, 2011). Thus, when a third party successfully installs a malicious content such as worms, Trojan horses, etc., on a smartphone platform, it compromises user's security and privacy or even gain complete control over the device. Mobile application originates from third parties with possible introduction of malicious

exploits such as utilization of computing power and manipulation of data stored. Hence the paper, looked at security threats from the attacker or perpetrators point of view using a proposed model which considers three (3) key or thematic issues: attacker's goals, attack vectors and mobile malware. The paper analyzed the attacker's goals and motives which include collection of private data targeting both the confidentiality and integrity of stored information, utilization of computing resources as do malicious actions which may range from data loss to draining the device battery (Racic, Ma and Chen, 2006).

The attack vectors considered the delivery methods and attack strategies. The paper categorized the multiple mobile platforms attack vectors into four namely; mobile network services, internet access, Bluetooth and access o USB and other peripheral devices. The third key issue, mobile malware is possible due to the resemblance of the traditional operating system and the mobile operating system. According to the paper, the notable malware includes; Trojan horse used to gather private information, Botnet, eg: Wlaedae (Flo and Josang, 2009) employs SMS and MMS messages to exchange data between nodes, worm; a self-replicating malicious application ported to mobile platforms through the introduction of cabir (Dunham, 2008) and Rootkit, a malicious application used to gain rights to run in a privileged mode.

Delac and colleagues suggested that, to address the security issues the Android platform implements a permission-based security model comprising of four types of permission: normal, dangerous, signature and signature-on-system (Delac, Silic and Krolo, 2011).

Normal permissions are granted without an explicit user's approval give access to isolated application-level functionalities with little impact on system or user security. Dangerous permission provides access to private data and critical systems hence considered a high security risk. Signature permission is extended to the application to signal if it is the same certificate as application declaring the permission whilst signature-or-system permission adds on to the signature permission by granting permission installed in the Android system image.

Contributing to the Android security issues, Yadav et al (2017) in their paper titled, "Android vulnerabilities and security" looked at the structure of an Android application consisting basically of four components: a single screen (Activities), background or remote processes (services), Apps data sharing (content provider) and Broadcast receivers which respond to a system wide broadcast announcement (Yadav *et al.*, 2017). The activities, services and broadcast receivers are activated by a message called Intent. The security architecture of Android which focused on sand boxing of applications was reviewed together with the role of content provider and Android permissions. In Android OS, an application process is a secure sandbox which is embedded in the OS to lessen the security

issues by isolating the applications data and codes from other application. The content provider is used to share private data between applications whilst the content

Resolver is used to communicate between client and the content provider (Nitya, 2017).

However, some security threats and vulnerabilities identified by the paper include,

1. the use of some reverse engineering tools such as Dedexer and Apk Tool could be used to disassemble an application source code, modify the source code and reassembly the source code to the original application thereby compromising the security. To prevent reverse engineering being adopted for malevolent purposes. The Android "Proguard" feature could be used.
2. Open sourced community contribution to security tends to increase the risk of attack hence the level of vulnerability
3. Verification of "malicious-free" applications in which installed application's extracted signature is compared to a repository of known malicious applications for commensurate action to be taken based on a match or otherwise. This procedure appears to be ineffective when such malicious application is not popular, or changes made to the signature of a known malicious application.

## **2.8 Forensic Analysis of Android Mobile Devices**

In this paper, Rao and Chakravarthy (2016) alluded to the fact that, there exist powerful features and technology available on mobile devices that runs the Android Operating

System which facilitates user activities such as exchange of text messages, voice calls, audio files, video files (Rao and Chakravarthy, 2016). Images, business transaction files etc. the paper accordingly stated that, diverse crimes such as harassment via SMS, fraud over e-mail, child trafficking and pornography, etc may be committed with technological features provided by the Android OS platform. Thus, in a nutshell, the smartphones are repository of large data volumes, which is extremely useful to analyst during an investigation. The paper reviewed other forensic activity on Android OS such as; gaining super user privileges, imaging required partitions using "dd" command through Android Debug Bridge (ADB) by (Lessad and Kessler, 2010), recovery booting method providing a repeatable and consistent collection of numerous Android devices in a normal operating mode without any "rooting: process by (Vidas, Zhang and Christin, 2011), fast imaging and analysis of data partition area of an Android device based on the yaffs file system where a wealth of information recovered in a forensically sound manner without any specific tool or system by (Aouad and Kechadi, 2011).

In line with the paper's objective of dealing with the techniques and tools to seize the suspected device in a forensically sound manner and perform the analysis thereof on the acquired forensic image, three-step procedure for the 4 acquisition and analysis of artifacts available in the Android device. The proposed steps are

- i. Rooting the Android device using “kingroot” software to get super user privileges, effective rooting and to avoid data loss and other problems.
- ii. Forensic acquisition of the device memory using the adb command terminal
- iii. Analysis of the acquired forensic images of partition such as /data/cache and/system are analysed using FTK Imager.

Amongst the data available in the partitions, the paper focused on obtaining data such as contacts, groups information, all history, SMS/MMS, browsing history, log records, email messages, installed apps and data, instant messaging applications and user accounts details enabled on the device, even though the enumerated focal sources in this paper could provide data with evidential value, the paper did not provide details or content of such data from the imaged partitions.

## **2.9 Android Digital Forensics: Data, Extraction and Analysis**

In this paper, Scrivens and Lin (2017) not only acknowledged the prevalence of smartphones notably those with Android Operating System (Scrivens and Lin, 2017) but, also recognized the surge of applications available on such devices which process a significant amount of personal information. According to Scrivens and Lin (2017), as smartphones become more prevalent in our society, there is a witness in their usage to perpetuate crimes (Scrivens and Lin, 2017). Hence, the assertion, smartphone device found

at crime scene are prime sources of evidence and requires forensic analysis. The aim of this paper is to provide answers for the following research questions;

- i. How much of user's private/personal data do these installed mobile applications store locally in the smartphone's storage that may be of significance to a forensic investigation?
- ii. How can these significance data be extracted?

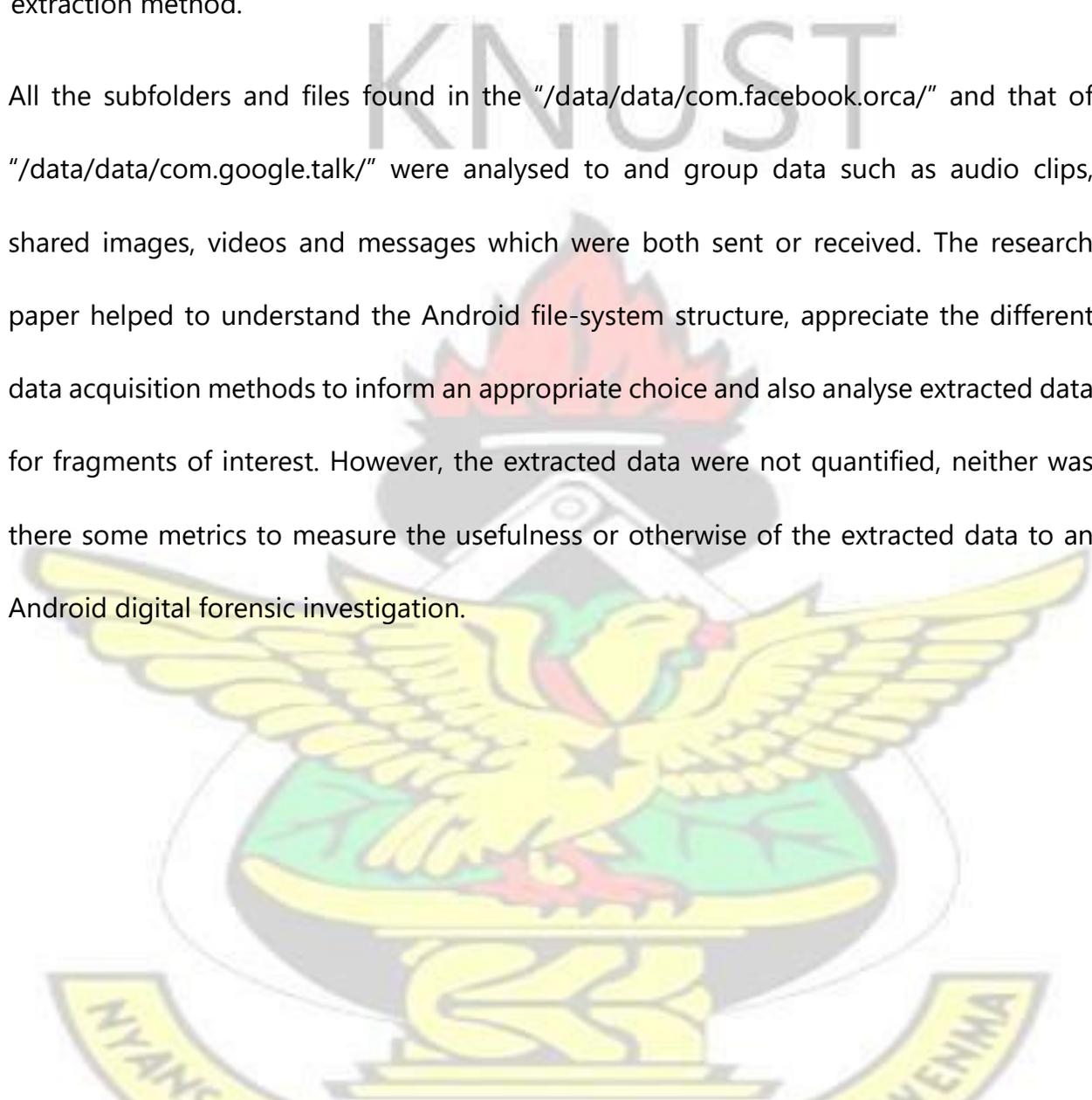
In the light of these research questions, the paper adopted a three-fold mobile device forensic investigation approach which comprises of; identification of data storage location(s), data extraction and data analysis. To practicalize these approaches, the paper reviewed the Android storage structure to identify possible storage locations where relevant data could reside. The notable storage locations or directions identified are "/data" where the "user data" partition which contains all private application storage are mounted, "/sdcard" where the "scard" partition containing useful information such as pictures, video, download files, etc are mounted and the "/system" directories. The scope of the paper was limited to the "/data" and "/sdcard" Android storage locations.

The paper further reviewed some data acquisition methods which are used to either make a logical or physical image of a storage device or partition. The notable data acquisition methods reviewed in the paper are clip-off and STAG (suitable for physical images), Android Debug Bridge (ADB) and Backup Applications (Forensic Software Suites) as well

as firmware update Protocol in which the device can be booted into firmware update or download mode and Custom Recovery Image.

The research was implemented by generating data through the installation and usage of Facebook Messenger and Google Hangouts for some appreciable amount of time. The data generated by the Facebook Messenger and Google Hangouts were located in `"/data/data/com.Facebook.orca/"` and `"/data/data/com.google.android.talk/"` respectively. The data was forensically retrieved using the custom Recovery Image data extraction method.

All the subfolders and files found in the `"/data/data/com.facebook.orca/"` and that of `"/data/data/com.google.talk/"` were analysed to and group data such as audio clips, shared images, videos and messages which were both sent or received. The research paper helped to understand the Android file-system structure, appreciate the different data acquisition methods to inform an appropriate choice and also analyse extracted data for fragments of interest. However, the extracted data were not quantified, neither was there some metrics to measure the usefulness or otherwise of the extracted data to an Android digital forensic investigation.

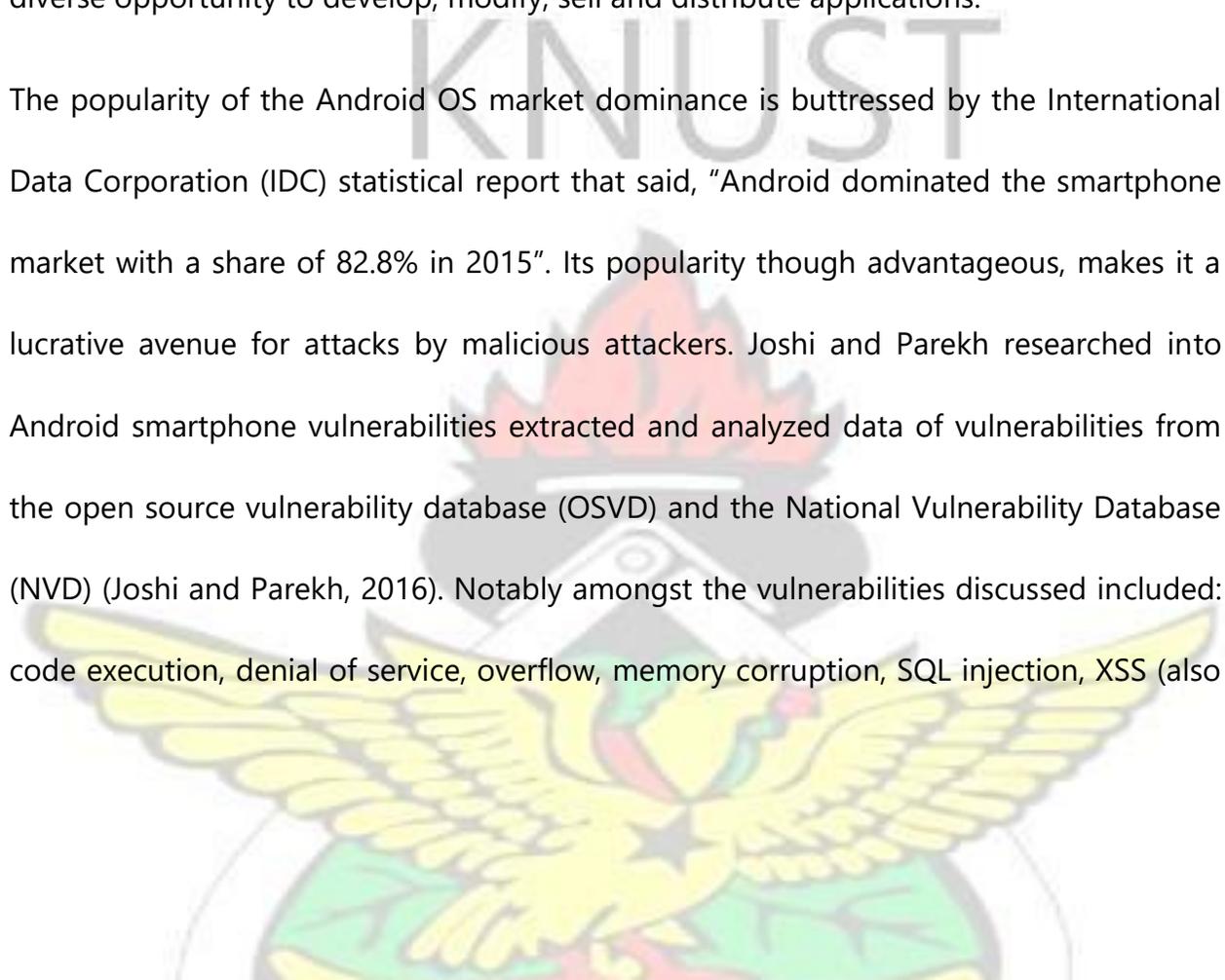


## 2.10 Vulnerabilities in Android Smartphones

Common Vulnerabilities and Exposures (CVE) defines a vulnerability as: "A weakness in the computational logic (e.g., code) found in software and hardware components that, when exploited, results in a negative impact to confidentiality, integrity, or availability".

**The Open Web Application Security Project (OWASP)** also be defined vulnerability as "a hole or a weakness in application, which can be a design flaw or an implementation bug, that that allows an attacker to cause harm to the stakeholders of an application" (OWASP, 2017). The penetrative ability of several smartphones may be largely attributed to the popularity and availability of the android operating system. The Android OS is an open source software which employs the Linux kernel as its base, affording developers diverse opportunity to develop, modify, sell and distribute applications.

The popularity of the Android OS market dominance is buttressed by the International Data Corporation (IDC) statistical report that said, "Android dominated the smartphone market with a share of 82.8% in 2015". Its popularity though advantageous, makes it a lucrative avenue for attacks by malicious attackers. Joshi and Parekh researched into Android smartphone vulnerabilities extracted and analyzed data of vulnerabilities from the open source vulnerability database (OSVD) and the National Vulnerability Database (NVD) (Joshi and Parekh, 2016). Notably amongst the vulnerabilities discussed included: code execution, denial of service, overflow, memory corruption, SQL injection, XSS (also



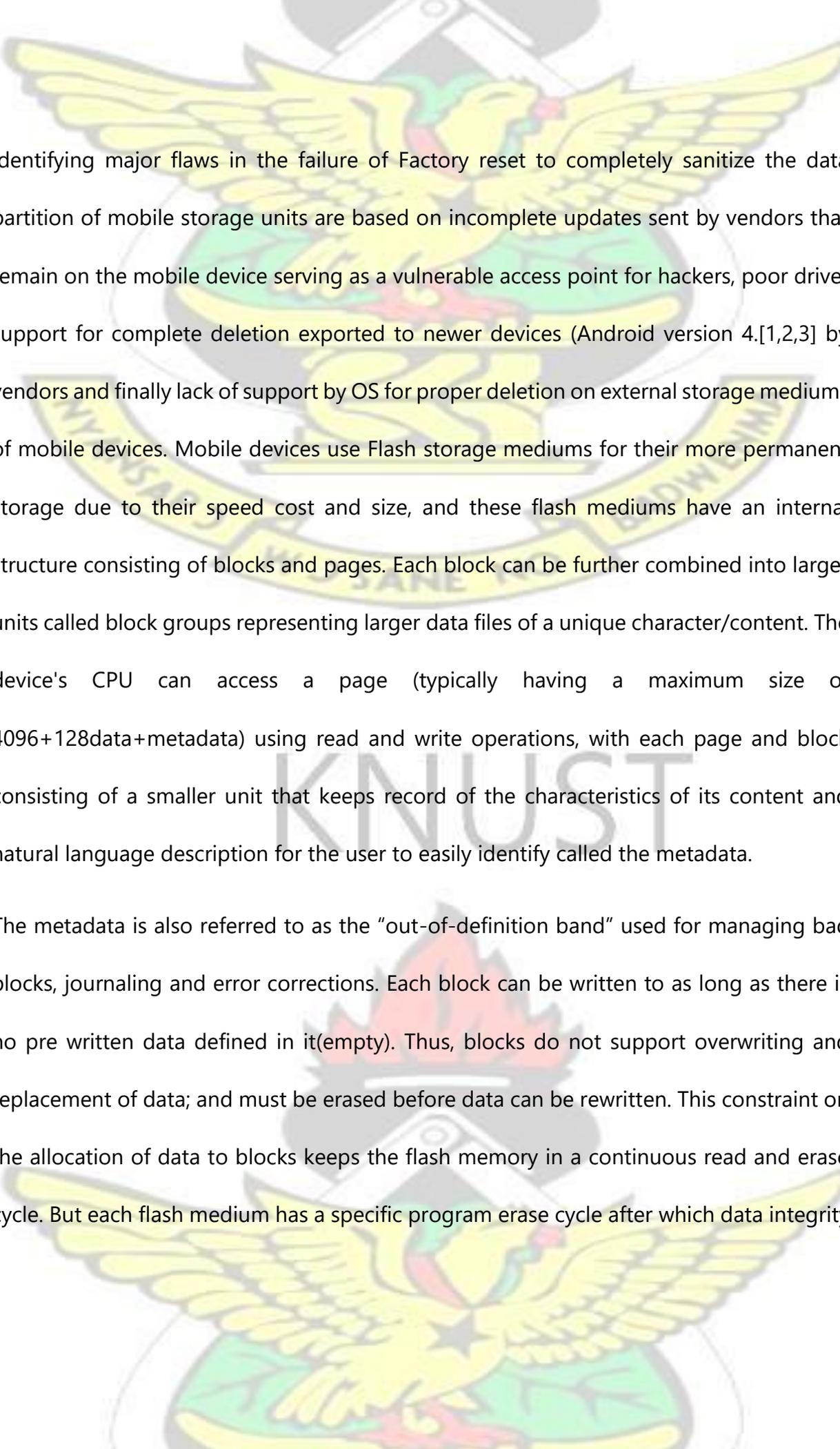
known as Cross Application Scripting), directing traversal, Http response splitting, bypass something, gain information/gain privilege, cross site request forgery (CSRF) and file inclusion.

After discussing the above enumerated vulnerabilities, a metadata model for the vulnerability was proposed. The proposed vulnerability metadata model could be likened to a tree data structure, with "Vulnerability" as it's root and the leaves are; Life Cycle, Severity, Category and Relation.

Though the research threw light on some prevailing vulnerabilities in Android smartphones and alluding to the fact that, mobile and Android devices were not particularly designed with privacy in mind. The research only proposed a model that was not implemented to illustrate how it can help reduce or improve on the vulnerabilities discussed in the research.

### ***2.10.1 Vulnerabilities of Factory Reset***

Large amount of user information relating to activities and transactions performed by the owner of the device have very high data retention rate even after Factory reset is performed on the specific device. The flaws in factory reset enables any individual with adequate knowledge in the internal functionalities and storage locations of android mobile devices to easily access Google accounts and associated backups of data performed by google services, such as wireless network information and contacts.

The logo of the Kenya National University of Science and Technology (KNUST) is centered in the background. It features a yellow eagle with its wings spread, perched on a green shield. Above the eagle is a white cross. Below the eagle is a yellow banner with the text 'KNUST' in black. The entire logo is set against a light blue background.

Identifying major flaws in the failure of Factory reset to completely sanitize the data partition of mobile storage units are based on incomplete updates sent by vendors that remain on the mobile device serving as a vulnerable access point for hackers, poor driver support for complete deletion exported to newer devices (Android version 4.[1,2,3] by vendors and finally lack of support by OS for proper deletion on external storage mediums of mobile devices. Mobile devices use Flash storage mediums for their more permanent storage due to their speed cost and size, and these flash mediums have an internal structure consisting of blocks and pages. Each block can be further combined into larger units called block groups representing larger data files of a unique character/content. The device's CPU can access a page (typically having a maximum size of 4096+128data+metadata) using read and write operations, with each page and block consisting of a smaller unit that keeps record of the characteristics of its content and natural language description for the user to easily identify called the metadata.

The metadata is also referred to as the "out-of-definition band" used for managing bad blocks, journaling and error corrections. Each block can be written to as long as there is no pre written data defined in it(empty). Thus, blocks do not support overwriting and replacement of data; and must be erased before data can be rewritten. This constraint on the allocation of data to blocks keeps the flash memory in a continuous read and erase cycle. But each flash medium has a specific program erase cycle after which data integrity

cannot be ensured. In order to alleviate this problem, wear-levelling algorithms are implemented to equally distribute erase and write operations over every block.

In the process of deleting files, the Operating system typically deletes the target data name from the index/ table rather than wiping the block of its content. On flash mediums, because data replacement or update does not occur, in order to maintain performance, new data is copied to a new block in order to evenly reduce the erasure block count and minimize wear.

Therefore, many recommended data sanitization standards are defined, each standard pertaining to the level of threat that is aimed to eliminate (Threat model) (Chang and Kuo, 2011). The highest level of these standards being analog procedure; which drastically

reduces the analog signals that encodes data in order to prevent reconstruction even with the most sensitive signal sensing equipment and expertise.

Digital procedure prevents data from being recovered by any digital signaling means, including altering or bypassing device controllers. Which would require datasheets for these firmware components which are not open to public.

Finally, logical procedure prevents recovery by any standard hardware interface commands (E.MMC commands). Thus, a safer and more preferred suggestion to preventing data retention is to delete the data manually before performing a factory reset (NIST 800-88's "clearing" level). But all these techniques are still vulnerable to high level recovery and carving techniques. In conclusion, the best alternative is to ensure that data recovered is rendered unintelligent to the hacker or analyst by form of encryption or data scattering techniques.

### **CHAPTER THREE**

### **METHODOLOGY**

### **3.0 Introduction**

This chapter describes how the study was conducted in three phases, beginning with a survey to understand the permeation of the Android operating system in the Ghanaian smartphone market as well as the brands that have the larger market share in the country. From there this chapter proceeds to describe how data can be extracted from used Android devices using open source forensic software tools, both before and after a factory reset (data persistence) has been carried out on the device. The chapter concludes with some proposed solutions to the problem of files being recoverable even after a factory reset has been carried out on an Android device.

### **3.1 Survey Work**

#### ***3.1.1 Study design and context***

A population-based, cross-sectional quantitative study was conducted to assess the community's knowledge and vulnerability awareness in the use of smartphone with Android OS in Ghana. The quantitative design ensured the independence of the researcher from the study findings and is also premised on the assumption that there is a social reality external to the knower and knowledge is objective and tangible (Patton, 2002). The study also drew on the empiricist paradigm (Creswell, 2003) in which the investigator can study a phenomenon without influencing it or being influenced by it;

“inquiry takes place as through a one-way mirror” (Guba and Lincoln, 1994). Thus, great emphasis is placed on ensuring validity of the research findings by the process of rigorous clarification and reliability using statistical tests (Milne, 1998). In view of this, large sample sizes are used to ensure representativeness of the study findings (Carey, 1993). The study instrument was divided into two sections. Section ‘A’ elicited data on the biodata of respondents: gender, age, educational level and current phone model used. Some of data sought from section ‘B’ include information usually accessed/stored on your phone, mode of disposing used phone, information considered to be private, mode of getting rid of private data before disposing of phone, knowledge about encryption of Android phones and means used to retrieve deleted files.

The study transcends the administrative boundaries of Ghana located at the center of the West African sub-region. With a total land size of 238,533 square kilometres, Ghana lies between Latitudes 4<sup>0</sup> and 12<sup>0</sup> North and Longitudes 4<sup>0</sup>W and 2<sup>0</sup> East. The country shares boundaries with three Francophonic countries: Cote D’ivoire to the west, Togo to the east, and Burkina Faso to the north, with the Gulf of Guinea and the Atlantic Ocean to the south (Ghana Statistical Service (GSS), 2013). One of Africa’s largest mobile market is Ghana with about 36.61 million subscribers as at the end of the second quarter of 2016(NCA, 2016).

### **3.1.2 Sample and Procedure**

Individuals aged 18 years and above were included in the study since this age is defined as the maturity age in Ghana where one is eligible to make informed decisions (Ghana Statistical Service (GSS), 2013). The need to identify respondents above 18 years, are within accessible geographical proximity, and are available and willing to participate in the study, led to the use of convenience sampling (Palinkas *et al.*, 2015). Etikan and colleagues argues that participants selected through convenience sampling techniques are not different from those drawn with random sampling techniques since the former is guided by the principle of homogeneity and ensures that knowledge gained is representative of the population (Etikan, Musa and Alkassim, 2016). Therefore, 1240 sampled respondents participated in the study. Respondents for the study were sampled from various households across the length and breadth of the country from October 2016 to December 2016. However, in order to ensure the reliability of the research instrument, the questionnaires (APPENDIX 1) and participatory activities were pre-tested with 150 Computer Science students at the Kwame Nkrumah University of Science and Technology (KNUST).

The researcher and his team, Research and Teaching Assistants from KNUST, invited these students to respond to the research instrument and questions through a general announcement in class. Statements which were not clear, not easily understood, in the questions were highlighted by the students and this informed the modification process.

The revised questionnaire was then used in the final administration to the general populace. This sought information on respondents' biodata, previous phone used and mode of disposal, predominant activities undertaken on the phone and knowledge about Android security and data recovery techniques and all outcome presented in tabular form. The responses of the pilot participants were not included in the final analysis. Responding to the questionnaire lasted an average of 25 minutes.

### ***3.1.3 Data management and analysis***

The edited data was captured in the Predictive Analytics SoftWare (PASW) version 16 and analysed using descriptive and inferential statistics. The descriptive statistics were presented by means of tables whereas some were converted into bar graphs using Microsoft excel 2013 (see Chapter 4). Since respondents used different models of smartphones (Android and non-Android), the data was split using the "split data" and "select cases" command in the PASW so that cases of respondents who use Android smartphones could be analyzed. The options for mode of phone disposal was further recoded from its initial five options (sold, gift, exchange, stolen/missing and still keeping) to two variables: sold, gift, stolen/missing and exchange were recoded as 'gave it out' whereas still keeping was maintained. Besides this, chi-square test analysis was used to examine relationship between the study variables. Also, linear logit model was used to predict respondents' knowledge on Android security and data recovery based on their

biodata. The underlying assumption needed to satisfy the use of both the chi-square test and regression analysis were assessed in order to ensure the validity of the result

(Vardeman and Morris, 2003).

### **3.1.4 Ethical consideration**

The purpose of the study was explained to all the respondents and were assured of the confidentiality and privacy of the information they provide. Participants were informed that the study is purely for academic purposes and that, participation was voluntary. Besides this, all the research team presented their identification cards as proof of their affiliation to the Department of Computer Science, Kwame Nkrumah University of Science and Technology (KNUST).

## **3.2 Android Data Persistence Analysis**

### **3.2.1 Introduction**

Design Methodology was adopted in this research. This methodology gives the researcher clearer understanding of the problem and also provides better inside answers to problem at hand while at the same time providing answers to the most fundamental questions faced throughout the process. It provides answers to questions like:

- What is the correct design?
- The characteristics it should have
- Is the solution working as intended?

The Design Research Methodology does not necessarily have to lead a solution though

but provide some good and feasible ideas (Lee, 2012).

The research was carried out using several common phases through conceptualization, verification, validation, design, implementation and testing.

The outcome of the methodology resulted in the design of the Headless Block Swapping (HBS)

The study set out to demonstrate that data on an Android device persists on the physical disk even after it appears to have been erased in a factory reset. The study used twenty (20) Android mobile devices (page 75) from different manufacturers to investigate the performance of different brands and Android versions when it comes to erasing data from the disk.

### **3.2.2 Overview of Analysis (Method)**

The Data Persistence Analysis was carried out as follows:

- An image of the device's internal memory was extracted using open source tools.
- The device was subjected to a factory reset in order to wipe data from the device.
  - A second image of the device was extracted using the same procedure as before.
- Two file carving tools were used on both images to acquire files, both the files that are "deleted" and those that are not.

- The number of files carved before the devices were factory reset was compared • 8 with the number of files carved after the factory reset.

### 3.2.3 Image Acquisition Tools

Several hardware and software tools were employed for the purpose of obtaining copies of the internal memory from the Android devices. Open Source software was preferred where available.

### 3.2.4 Hardware and Software Tools

The study used the following Hardware and software tools

#### Hardware Tools:

- 64-bit Ubuntu 16.04 LTS and
- 64-bit Kali Linux operating systems installed on
- Two (2) HP EliteDesk 800 G1 TWR desktop computers with
- Intel® Core™ i7-4790 CPU running at 3.60GHz × 8,
- 12 GiB memory and 1 TB hard disk,
- 64-bit Windows® 7 Professional operating system running on a
- Dell Precision 3620 desktop computer with • Intel® Core™ i7-6700 CPU running at 3.40GHz,
- 12 GiB memory and 1 TB hard disk.

- 20 used Android smartphones and tablet computers from various manufacturers, running different versions of the Android operating system. The brands selected were the ones which had the most users in Ghana according to the survey conducted in this study. Table 3.1 shows the devices that were used by the study.

Table 3.1- Summary of devices that were used for the data persistence analysis

<b>No.</b>	<b>Brand</b>	<b>Model</b>	<b>Android Version</b>
1	Samsung	A5 (3GB RAM, 32GB ROM)	Android 7.0 – Nougat
2	Samsung	S7 (4GB RAM, 32GB ROM)	Android 7.0 – Nougat
3	Samsung	A8 (4GB RAM, 64GB ROM)	Android 7.0 – Nougat
4	Samsung	Galaxy Note 4 (3GB RAM,32GB ROM)	Android 4.4.4 -KitKat
5	Samsung	Galaxy Note 5 (4GB RAM, 32GB ROM)	Android 5.1 – Lollipop
6	Itel	A11 (512MB RAM, 8GB ROM)	Android 6.0 – Marshmallow
7	Itel	it1 556 plus (1GB RAM, 8GB ROM)	Android 5.1 – Lollipop
8	Itel	S31 (1GB RAM, 16GB ROM)	Android 6.0 – Marshmallow
9	Itel	S12 (1GB RAM, 8GB ROM)	Android 7.0 – Nougat
10	Itel	S32 (1GB RAM, 16GB ROM)	Android 7.0 – Nougat
11	Infinix	Zero 4 4 (3GB RAM,32GB ROM)	Android 6.0 – Marshmallow
12	Infinix	S2 Pro (3GB RAM,32GB ROM)	Android 6.0 –

				Marshmallow
13	Infinix	Zero 5 (6GB RAM, 64GB ROM)	Android	7.0 – Nougat
14	Infinix	Zero 3 (3GB RAM,32GB ROM)	Android	5.1 – Lollipop
15	Infinix	Hot 2 (2GB RAM,16GB ROM)	Android	5.1.1
16	Techno	W5 (2GB RAM,32GB ROM)	Android	6.0 – Marshmallow
17	Techno	L9 (3GB RAM, 16GBROM)	Android	6.0 – Marshmallow
18	Techno	Camon CX (3GB RAM, 16GB ROM)	Android	7.0 – Nougat
19	Techno	Spark K7 (1GB RAM, 16GB ROM)	Android	7.0 – Nougat
20	Techno	Phantom 8 (6GB RAM, 64BG ROM)	Android	7.0 – Nougat

---

**The software tools used by the study are as follows:**

- Kingo Root Windows® application: for rooting the android phones. This tool is required since the Android devices need to be operated in Superuser mode in order to access low level functions such as reading data directly from the physical disk.
- BusyBox Android app for installing the BusyBox binary which gives access to several Linux operating system utilities such as NetCat which is required for transferring data from the smartphone or tablet to the computer. This tool is required since the

Android operating system by default doesn't come with these tools installed. The operating system was designed to be used from the graphical user interface rather than the command line, hence the omission of the tools which BusyBox provides.

- NetCat \*(NC) command-line utility. This utility allows for network communications from the command line. It comes pre-installed on both Kali Linux and Ubuntu. This tool is required for transferring a large amount of data over a network connection between the Android device and the computer being used to extract the data. Since the size of the data being extracted is the same size as the Android device's internal memory, it is impossible to store the file on the device before transferring over USB.
- Foremost Linux command-line file carving utility. This utility is installed by default on Kali Linux but has to be installed from the standard repository on Ubuntu. Foremost is a file carving tool, designed to operate from the Linux environment. This tool is required for performing the low-level file carving functions needed by the study.
- Scalpel Linux command-line file carving utility. This utility is installed by default on Kali Linux but has to be installed from the standard repository on Ubuntu. Scalpel is another carving tool, built as an improvement on Foremost. Scalpel was included in the study to give a fuller view of what the open source tools would be able to extract from the Android devices.
- Android Debug Bridge (ADB) command-line utility. The ADB utility allows a

computer to communicate with an Android device via USB. It has to be installed from the terminal on both Kali Linux and Ubuntu. This tool is required for establishing a connection to the lower level processes and functions of the Android device. ADB allows a computer to access functions on the Android device which are mostly hidden from regular Android users who mainly use the graphical user interface.

- DB Browser/SQLite Browser GUI application. This utility allows for opening SQLite files to view their contents. There are versions for Windows and Linux. It comes pre-installed on Kali Linux but must be installed from the Software Center in Ubuntu or downloaded and installed on Windows. This tool is required for viewing SQLite database files to determine which ones contain data on the user's SMS and contacts.

All the listed software is free to download and use, and with the exception of Kingo Root, they are all open source software.

### **3.2.5 Image Acquisition Process**

The steps undertaken in acquiring an image of the Android devices' internal memory are outlined below.

- Superuser access on the Android device was acquired using the Kingo Root

application. This process is commonly referred to as "Rooting".

- The BusyBox application was installed on the device from the Google Play Store.
- From the BusyBox application, the BusyBox binaries were installed by tapping the "Install BusyBox" button.
- The DD command was used to read the device's internal memory into a buffer in memory.
- The data in the memory buffer was piped through the BusyBox NetCat utility to one of the device's TCP ports.
- From the computer, data was read from the device's TCP port over an ADB connection, and written to the computer's larger disk

Since Superuser access persists after a factory reset, the first step was excluded when acquiring an image from a device for the second time.

Each device used for the study was imaged twice, once before and another after a factory reset was performed on the device. The steps in the image acquisition process are expanded in the subsections that follow.

### **3.2.6 Rooting the Device**

For Unix-like operating systems, complete access to all the system's functions and data is

available to a single type of user, known as the Root User or Superuser. By default, other users of the operating system have various restrictions placed on what they can do and which files they can access. Most Android devices do not give users the root access out of the box since people who do not understand the basic operations of the operating system may alter the system in a manner that renders the device unusable. The process of rooting an Android device refers to the use of various software exploits or hacks to make root access available on the device. While in the past, rooting an Android device was a relatively dangerous exercise involving manually running a number of exploits to find which one works, in modern times "One-click" rooting applications are available which make rooting and un-rooting Android devices relatively safer and much easier to perform.

Prior to imaging an Android device, root access must first be acquired so as to allow the researchers to access sensitive and potentially dangerous operating system commands and files. The study employed the Kingo Root Windows® GUI application as a one-click solution to root the Android devices.

Figure 3.1 (Appendix 1) shows the default interface for the Kingo Root application. For the Kingo Root application to successfully communicate with the Android device, the device must be connected to the computer in Android Debug Bridge (ADB) mode using a USB data cable and also, ADB drivers for the device must be installed on the computer.

The ADB drivers for each device can be obtained from the manufacturer's support section on their website. Otherwise, a universal ADB driver can also be acquired online for the

same purpose.

To activate Android Debugging Bridge (ADB) mode on an android device, the means of reaching the settings differ from device to device and for devices running Android 4.2 or later, the section of device settings which deals with Developer Options is hidden by default and must be activated by going to:

Settings >> About Device >> Build Number

Figure 3.2 (Appendix 1) shows the Android Settings application.

Figure 3.3 (Appendix 1) shows the "About Phone" section of the Android Settings app with the "Build Number" list item being tapped on six (6) times and a notification showing that "Developer Options" will be enabled soon.

Figure 3.4 (Appendix 1) shows the "About Phone" section of the Android Settings app with "Developer Options" enabled.

By tapping on "Build number" 7 times, the Developer Options section of the settings is enabled. In general, entering ADB mode on an Android device involves enabling the Android debugging setting at:

Settings >> Developer Options >> Android debugging

Figure 3.5 (Appendix 1) shows the "Developer Options" section of the Android Settings

App. Turning the switch to “On” position causes a dialogue to appear warning of the potential danger of enabling the advanced options that are meant to be used by developers.

Figure 3.6 (Appendix 1) shows the Developer Options section with USB Debugging being enabled. Attempting to turn on USB Debugging option causes a dialogue to appear warning of the potential dangers.

After connecting the Android device to the computer in ADB mode, the Kingo Root application detects the device and shows a button for rooting the device. After clicking the button, the Kingo Root application connects to the internet to fetch known exploits that work on the device and proceeds to attempt rooting the device using the exploits found. When rooting is complete, the application notifies the user.

Figure 3.7 (Appendix 1) shows the screen of the Android phone after a connection has been established with Kingo Root desktop application.

Appendix 1)

Figure 3.8 ( shows the Kingo Root interface after a connection has been

## Appendix 1)

established with the Android device.

Figure 3.9 (Appendix 1) shows the Kingo Root desktop application in the process of rooting an Android device.

Figure 3.10 (Appendix 1) shows the Android phone screen while rooting with Kingo Root is in progress.

Figure 3.11 (Appendix 1) shows the Kingo Root desktop application after a successful rooting process.

Figure 3.12 (Appendix 1) shows the Android Phone screen after a successful rooting by King Root application.

### **3.2.7 Busybox Installation**

After gaining root access, the next step is to install BusyBox Android app on the device.

BusyBox Android app allows the installation of the BusyBox binary which provides the functions of several common Linux command-line utilities, one of which is netcat. Netcat is used for reading from and writing to network connections using TCP or UDP. It is required for transferring the image of the device's embedded secondary storage from the

## Appendix 1)

Android device to the computer. BusyBox is installed from the Google Play Store.

## Appendix 1)

Figure 3.13 ( ) shows the Busybox application being installed from the Google Play Store. The BusyBox binary is installed into the device's system binaries directory by clicking the "Install" button at the bottom of the BusyBox main activity.

### **3.2.8 Copying the Device Internal Memory**

Obtaining the image of the Android device was done from the Terminal in either Ubuntu or Kali Linux. The steps for extracting the image are outlined:

1. Root the Android device.
2. Install BusyBox Android app. When the app is launched, a popup appears with a request for Superuser permission. The Superuser permission allows the app to install application binaries into the core of the operating system from where they can be run with root privileges. This permission must be granted in order for the app to work.
3. Install the BusyBox binary from the BusyBox app.
4. Connect the Android device to the computer with ADB mode enabled on the device. If the device prompts for permission for the computer to communicate with the device via ADB, the permission must be granted.
5. Open two (2) Terminal windows on the computer running either Ubuntu or Kali Linux.

Appendix 1)

6. In the first Terminal window, obtain access to the device's command-line interface

Appendix 1)

using the command

```
>>$ adb shell
```

Figure 3.14 ( shows the output from the “ADB shell” command

7. In the same window, obtain root access to the device’s command-line interface using the command:

```
>>$ su
```

Figure 3.15 (Appendix 1) shows the output from running the “SU” command in “ADB Shell”

8. In the same window, obtain a list of the disks and partitions on the device using the command:

```
>># cat /proc/partitions
```

Figure 3.16 (Appendix 1) shows the output from running the “CAT” command

From the list, the name of the disk can be identified as the list entry with the largest corresponding size while the entries with smaller sizes are partitions of the disk. The names of the partitions usually begin with the name of the disk, followed by the letter

Appendix 1)

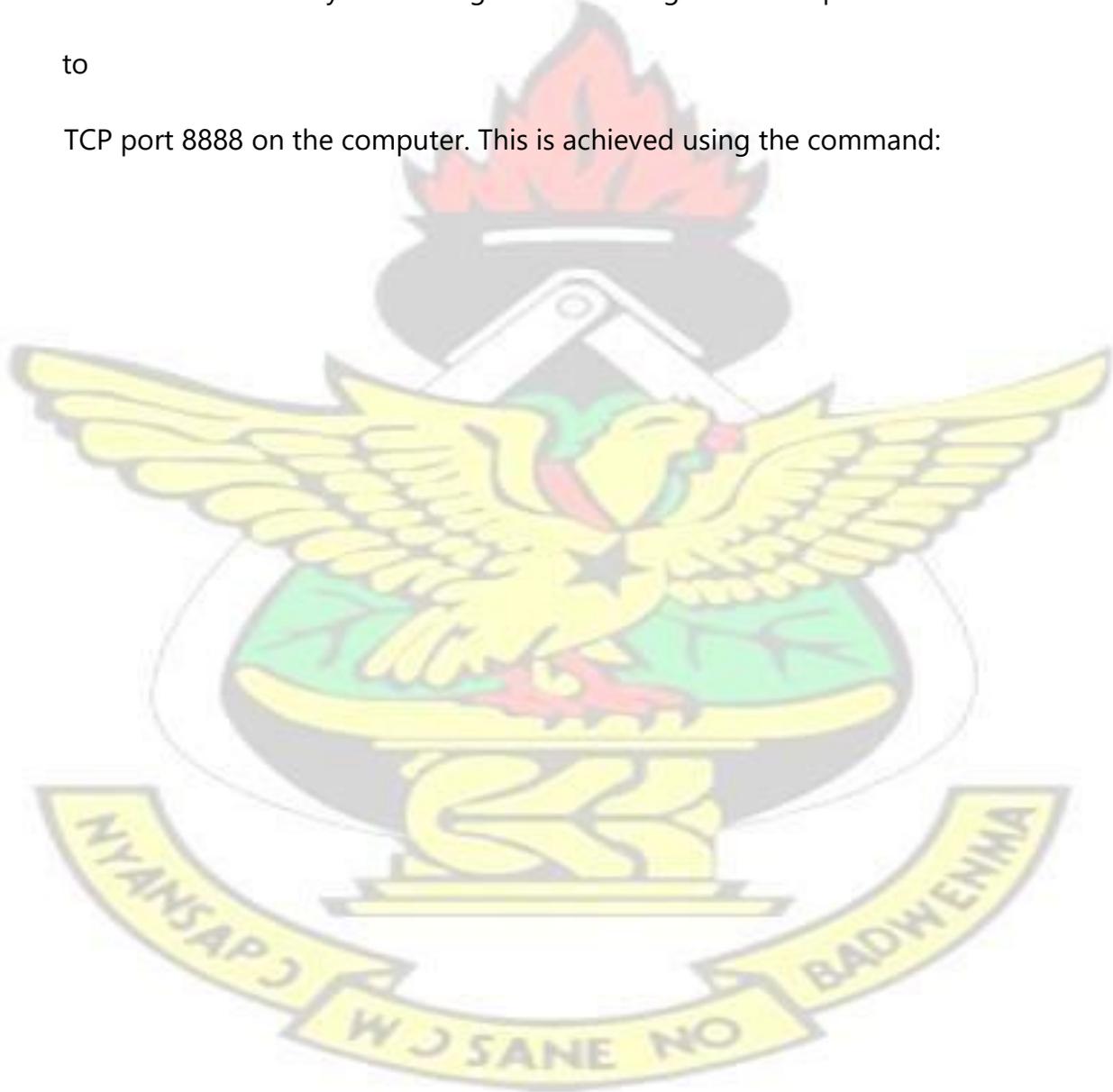
'p' then the partition number. For instance, if the disk name is mmcblk0, there are

Appendix 1)

partition names like mmcblk0p0, mmcblk0p1, mmcblk0p2 etc.

9. In the second Terminal window, open a TCP connection between the computer and the Android device by forwarding all data that goes to TCP port 8888 on the device to

TCP port 8888 on the computer. This is achieved using the command:



```
>>$ adb forward tcp:8888 tcp:8888
```

Aside from port 8888, any other port that is not currently in use and has not been reserved for another application can be used.

10. In the first Terminal window, read the internal disk using the dd command and pipe the output from the reading through BusyBox's implementation of the Linux netcat command to TCP port 8888 of the device. This is achieved using the command:

```
>># dd if=/dev/block/mmcblk0 | busybox nc -l -p 8888
```

Figure 3.17 (Appendix 1) shows the "dd" command executing and piping its output through the "nc" command

The parameter supplied to the dd command is the path to the device's internal disk pointer.

The parameter changes based on the name of the device's internal disk, as found in step 8 above. In the command just used, the name of the internal disk is mmcblk0.

11. In the second Terminal window, read data from the computer's TCP port 8888 into a file, which will be the image of the device's internal disk. This is achieved using the command:

```
>>$ nc 127.0.0.1 8888 > image.dd
```

Figure 3.18 (Appendix 1) shows the "NC" command being used to channel data from TCP port 8888 to a file

After completing steps 10 and 11 above, the Terminal windows will show no action while the

data transfer from the device to the computer is in progress. When the transfer is complete, the Terminal windows will show readiness to accept commands. The image file will be saved to the current working directory of the second Terminal.

### **3.3 FILE CARVING PROCESS**

Each extracted image was subjected to file carving using two Open Source file carving tools and the number of files carved was recorded. The two carving tools used by the study are Foremost and Scalpel. The subsections that follow go into the details of how the file carving was carried out.

#### **3.3.1 Installation of File Carving Tools**

1. Foremost: Foremost was installed from the standard repository on Ubuntu using the command shown:

```
>>$ sudo apt-get install foremost
```

Figure 3.19 (Appendix 1) shows the installation command for foremost and the output in an Ubuntu terminal.

2. Scalpel: This utility is installed from the standard repository using the command shown:

```
>>$ sudo apt-get install scalpel
```

Figure 3.20 (Appendix 1) shows the installation command for scalpel and its output in a Ubuntu terminal.

### 3.3.2 File Extraction and Enumeration with Scalpel and Foremost

Extraction of data from the disk image acquired from the dd command is done using any one of the two file carving tools, foremost or scalpel. To use either tool, a configuration file that is located in /etc directory can be modified to specify which files to search for and how to identify those files. The configuration for foremost can be found in /etc/foremost.conf and that of scalpel can be found in /etc/scalpel/scalpel.conf. To edit either configuration file, it must be opened in a text editor of one's choice with elevated privileges. For example, to edit the foremost configuration, the following command can be entered from the Terminal:

```
>>$sudo gedit /etc/foremost.conf
```

Figure 3.21 (Appendix 1) shows the foremost configuration file.

The contents of the configuration files for foremost and scalpel are identical since scalpel is based off foremost. In both cases entries are delimited with a new line character; i.e. each entry occupies a single line of its own. Each entry describes how to identify a file type by the header and footer and footer pattern of that file type. For each file type, the configuration file describes the file's extension, whether the header and footer are case sensitive, the maximum file size, and the header and footer for the file. The footer field is optional, but header, size, case sensitivity, and extension are not. To comment out an entry, the character "#" should be placed at the beginning of the line on which that entry

is recorded. Foremost and scalpel skip all entries that have the “#” character at the beginning of the line.

An entry for the mp3 file type would look like this:

```
mp3  y  8000000 \x57\x41\x56\45  \x00\x00\xFF\
```

The header and footer sequences can be expressed as plain text, in octal notation or in hexadecimal notation. Obtaining the header and footer sequences for a particular file type can be done by running a hexdump on a valid file of the desired file type and piping the output through the head or tail command. To obtain the header and footer of an mp3 file by name test.mp3, the following commands should be executed from the Linux Terminal:

```
>>$ hexdump test.mp3 | head
```

```
>>$ hexdump text.mp3 | tail
```

Figure 3.22 (Appendix 1) shows the “hexdump” command being used in conjunction with “head” command.

There are multiple entries in the configuration file that correspond to the same file type because of the various encodings used for the same file type. For instance, there are multiple entries for the mp3 file type due to the various encoding approaches used to create mp3 files.

In the default configuration files for foremost and scalpel, all the preset entries are

commented out. Hence to use the file carving tools, the default configuration has to be edited.

To extract data from a disk image using foremost, a similar command to the following must be executed:

```
>>$ foremost -o output_directory/ image_file.dd
```

Figure 3.23 (Appendix 1) shows the output from running the foremost tool to carve data from an image file.

To extract data from a disk image using scalpel, a similar command to the following must be executed:

```
>>$ scalpel -o output_directory/ image_file.dd
```

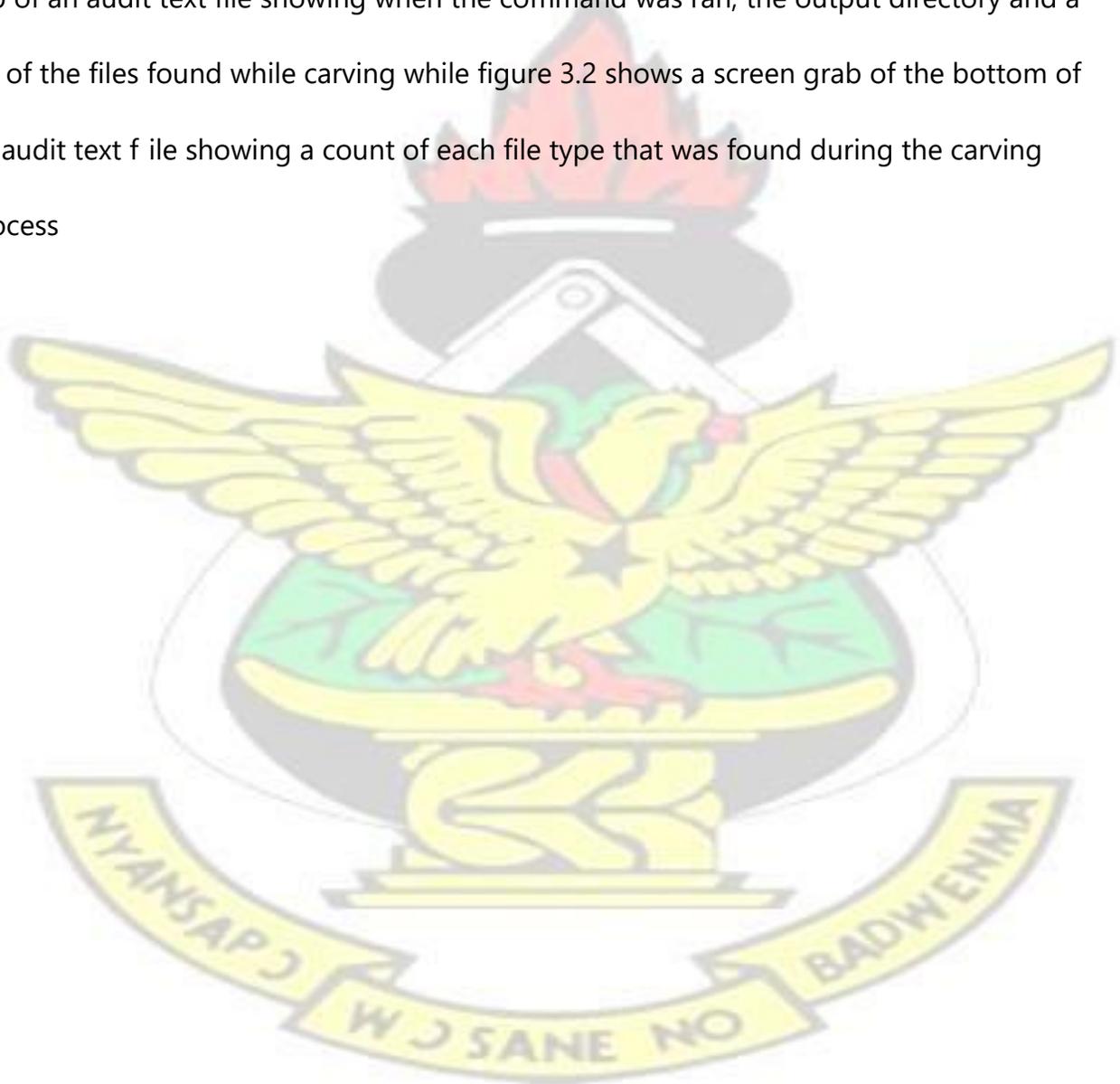
Figure 3.24 (Appendix 1) shows the output from running the scalpel carving tool on an image

The term "File Enumeration" as used in this document refers to the listing and counting of all files found during the file carving process by either foremost or scalpel. The listing and counting of files are a standard report generated by the carving tools to summarize the outcome of running the carving commands.

file.

After running the file carving program, the output of the carving is saved in the specified

output directory together with an audit text file that summarizes the results of the carving. The audit text file shows a complete list of all the files that were carved as well as giving a count of the files found according to the file type. Figure 3.1 shows a screen grab of the top of an audit text file showing when the command was ran, the output directory and a list of the files found while carving while figure 3.2 shows a screen grab of the bottom of an audit text file showing a count of each file type that was found during the carving process



```

audit7.txt
~/Downloads/Audit.txt
Save
Foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus
Audit File

Foremost started at Thu Mar 22 09:21:23 2018
Invocation: foremost -w -o before/ -i before.dd
Output directory: /home/research/Desktop/Image Extraction/tecno_y4/before
Configuration file: /etc/foremost.conf
-----
File: before.dd
Start: Thu Mar 22 09:21:23 2018
Length: 5 GB (5572132864 bytes)

Num      Name (bs=512)      Size      File Offset      Comment
0:       121194.jpg         2 KB      62051564
1:       177495.jpg        213 KB    90877616
2:       181135.jpg        20 KB     92741284
3:       181175.jpg        11 KB     92761880
4:       182937.jpg        43 KB     93664228
5:       50546.zip          72 MB     25879646
6:       50546.png          1 KB     25879740      (59 x 59)
7:       180564.png          1 KB     92448984      (63 x 63)
8:       180566.png        213 B     92450136      (12 x 12)
9:       180567.png        213 B     92450432      (12 x 12)
10:      180567.png         9 KB     92450728      (390 x 390)
11:      180587.png        287 B     92460608      (24 x 24)
12:      180587.png        305 B     92460984      (48 x 48)
13:      180588.png        548 B     92461380      (48 x 48)
14:      180589.png        621 B     92462020      (48 x 48)
15:      180591.png        752 B     92462732      (48 x 48)
16:      180592.png          1 KB     92463580      (41 x 41)
17:      180596.png          1 KB     92465224      (41 x 41)
18:      180599.png        229 B     92467068      (36 x 36)
19:      180600.png        225 B     92467396      (48 x 48)
20:      180601.png        280 B     92467712      (36 x 36)
21:      180601.png        583 B     92468080      (36 x 36)
22:      180603.png        359 B     92468756      (36 x 36)

Plain Text Tab Width: 8 Ln 1, Col 1 INS

```

Figure 3.1: Screen shot of top section of the audit text file from a foremost file carving command.

```
audit7.txt
~/Downloads/Audit.txt
Save

40802: 10710647.png 49 KB 5483851468 (640 x 640)
40803: 10710647.png 49 KB 5483851468 (640 x 640)
40804: 10752006.png 2 KB 5505027381 (50 x 50)
40805: 10752013.png 2 KB 5505030809 (50 x 50)
40806: 10752028.png 3 KB 5505038757 (50 x 50)
40807: 10752036.png 3 KB 5505042861 (50 x 50)
40808: 10752044.png 3 KB 5505046965 (50 x 50)
40809: 10752054.png 2 KB 5505051783 (50 x 50)
40810: 10752061.png 2 KB 5505055596 (50 x 50)
40811: 10752070.png 2 KB 5505059968 (50 x 50)
40812: 10752086.png 2 KB 5505068438 (50 x 50)
40813: 10752092.png 2 KB 5505071254 (50 x 50)
40814: 10752102.png 2 KB 5505076332 (50 x 50)
40815: 10752109.png 2 KB 5505080055 (50 x 50)
40816: 10752117.png 2 KB 5505084136 (50 x 50)
40817: 10752125.png 3 KB 5505088005 (50 x 50)
40818: 10860497.jpg 3 KB 5560574823
40819: 10879816.jpg 34 KB 5570465974
40820: 10880292.jpg 74 KB 5570709854
Finish: Thu Mar 22 09:31:26 2018

40821 FILES EXTRACTED

jpg:= 8298
gif:= 308
bmp:= 1
wmv:= 13
mov:= 15
mp4:= 21
rif:= 18
htm:= 129
ole:= 1
zip:= 144
png:= 31818
pdf:= 55
-----
Foremost finished at Thu Mar 22 09:31:26 2018|
Plain Text Tab Width: 8 Ln 40854, Col 46 INS
```

Figure 3.2: Bottom section of the audit text file from a foremost file carving command.

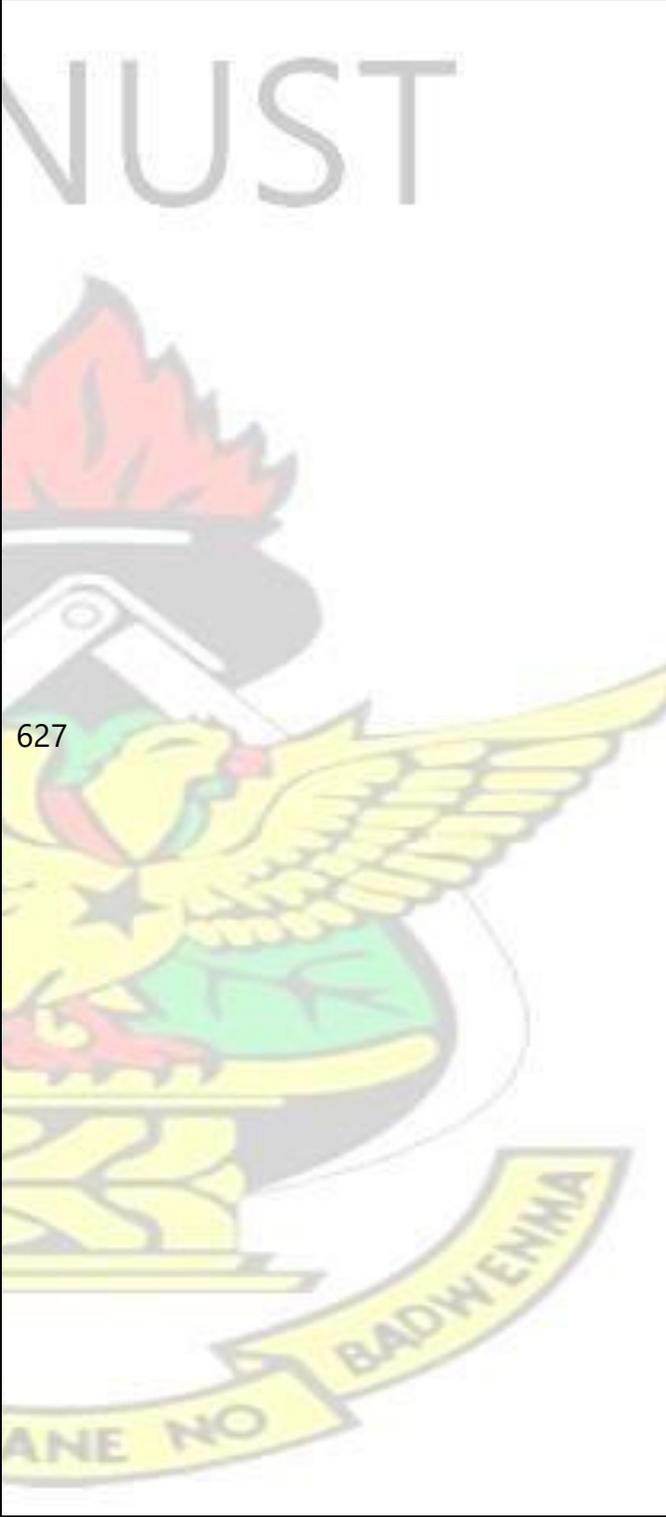
### 3.3.3 File Enumeration Using Scalpel

This section of the study presents the various counts of a few of the file types that were carved out using scalpel. The most popular file types for images, videos, audio, text and database files, are shown in the tables in the following sections. The term "File Enumeration" is used here to refer to the count of each file type, that was found by running the file carving commands on the Android devices that were used for the study. For the purpose of gauging how much data is retained after a factory reset, a file carving command was running on each Android device's image before the device was reset and afterwards another file carving command was ran again to get two counts for each file type, before and after factory reset.

The data enumeration results for Scalpel are shown in tables 3.2 and 3.3. Table 3.2 shows the enumeration results from the images that were extracted before factory reset while table 3.3 shows the enumeration results from the images that were extracted after factory reset.

Table 3.2: Data enumeration results from image files before the devices were reset, using Scalpel.

No./ID	JPEG	PNG	MP4	MP3	PDF	MS Word	SQLite	Gif
1. Samsung A5 (3GB RAM, 32GB ROM)	3724	597	78	40	303	2	159	43
2. Samsung S7 (4GB RAM, 32GB ROM)	3988	807	91	93	351	7	86	17
3. Samsung A8 (4GB RAM, 64GB ROM)	6750	207	53	47	327	39	172	30
4. Samsung Galaxy Note 4 (3GB RAM, 32GB ROM)	4720	112	27	3	359	47	129	4
5. Samsung – Galaxy Note 5 (4GB RAM, 32GB ROM)	7962	968	51	115	369	26	107	33

6. Itel A11 (512MB RAM , 8GB ROM)	345 6	 <p>627</p>	7	124	248	20	155	22
7. Itel it1 556 plus (1GB RAM ,	947 8	198	32	4	17	24	26	48

8GB ROM)									
8. Itel S31 (1GB RAM , 16GB ROM)	298 2	391	9	56	303	3	70	33	
9. Itel S12 (1GB RAM , 8GB ROM)	450 5	676	5	79	309	24	184	18	
10. Itel S32 (1GB RAM , 16GB ROM)	574 3	297	22	84	174	18	137	13	
11. Infinix Zero 4 (3GB RAM,32G B ROM)	229 9	655	45	118	58	24	183	6	
12. Infinix S2 Pro (3GB RAM,32G B ROM)	982 5	451	91	35	107	48	151	3	

13. Infinix Zero 5 (6GB RAM, 64GB ROM)	6523	149	95	7	59	27	56	44
14. Infinix Zero 3 (3GB RAM,32GB ROM)	2508	156	68	86	95	18	135	39
15. Infinix Hot 2 (2GB RAM,16GB ROM)	7066	444	30	120	141	41	178	44
16. Techno W5 (2GB RAM,32GB ROM)	8304	298	16	4	113	10	16	23
17. Techno L9 (3GB RAM, 16GBROM)	4049	141	40	23	335	16	83	13
18. Techno Camon CX (3GB RAM, 16GB ROM)	8829	740	16	94	246	43	185	17
19. Techno Spark K7 (1GB RAM , 16GB ROM)	6908	228	49	6	249	37	176	31
20. Techno Phantom 8 (6GB RAM , 64BG ROM)	7224	362	58	4	87	14	93	20

Table 3.3 – Data enumeration results from image files after the devices were reset, using Scalpel.

No./ID	JPEG	PNG	MP4	MP3	PDF	MS Word	SQLite	Gif
1. Samsung A5 (3GB RAM, 32GB ROM)	1771	280	37	18	143	11	56	18
2. Samsung S7 (4GB RAM , 32GB ROM)	1800	397	42	45	192	4	34	7
3. Samsung A8 (4GB RAM , 64GB ROM)	3300	97	22	19	161	20	83	14
4. Samsung Galaxy Note 4 (3GB RAM,32GB ROM)	2234	55	13	13	167	22	63	2

5. Samsung – Galaxy Note 5 (4GB RAM , 32GB ROM)	3730	456	23	58	179	12	50	16
6. ITEL A11 (512MB RAM , 8GB ROM)	2050	375	4	73	148	12	91	14
7. ITEL it1 556 plus (1GB RAM , 8GB ROM)	5953	117	20	7	10	16	17	32
8. ITEL S31 (1GB RAM , 16GB ROM)	1948	258	6	38	200	9	47	22
9. ITEL S12 (1GB RAM , 8GB ROM)	2791	427	3	49	192	15	117	11
10. ITEL S32 (1GB RAM , 16GB ROM)	3933	200	15	57	119	12	94	9
11. INFINIX Zero 4 (3GB RAM,32GB ROM)	1300	360	25	65	32	13	100	3
12. INFINIX S2 Pro (3GB RAM,32GB ROM)	5268	241	50	18	56	25	80	12

13. Infinix Zero 5 (6GB RAM, 64GB ROM)	3404		82	54	4	32	16	30	26
14. Infinix Zero 3 (3GB RAM, 32GB ROM)	1450		86	37	47	57	10	76	24



ROM)									
17. Techno L9 (3GB RAM, 16GBROM)	2504		85	25	16	230	10	61	10
18. Techno Camon CX (3GB RAM, 16GB ROM)	5531	393		11	64	169	31	128	12
19. Techno Spark K7 (1GB RAM , 16GB ROM)	5021	174		31	4	180	24	121	12
20. Techno Phantom 8 (6GB RAM , 64BG ROM)	5046	242		37	2	57	10	58	13

### 3.3.4 FILE ENUMERATION USING FOREMOST

Like what was done with the Scalpel carving tool, the count of each file type that was found while running the Foremost command, was taken for comparison. As before, the count was recorded before the device was factory reset and after the device was factory

reset. This was done to obtain a measure of the number of files that were retained despite a factory reset.

The data enumeration results for Foremost are shown in table 3.4 and 3.5. Table 3.4 shows the results from before the devices were reset and table 5 shows the results from after the devices had been reset.

Table 3.4 – Data enumeration results from image files before the devices were reset, using Foremost

No./ID	JPEG	PNG	MP4	MP3	PDF	MS Word	SQLite	Gif
1. Samsung A5 (3GB RAM , 32GB ROM)	3808	108	188	107	197	19	167	43
2. Samsung S7 (4GB RAM , 32GB ROM)	3460	336	98	59	90	1	106	31
3. Samsung A8 (4GB RAM , 64GB ROM)	5115	879	88	66	25	38	159	50
4. Samsung Galaxy Note 4 (3GB RAM,32GB ROM)	5475	270	40	28	159	28	5	43
5. Samsung – Galaxy Note 5 (4GB RAM , 32GB ROM)	7929	802	37	116	30	40	123	43
6. Itel A11 (512MB RAM , 8GB ROM)	3519	324	20	137	48	26	154	29
7. Itel it1 556 plus (1GB RAM , 8GB ROM)	5271	341	90	8	29	14	185	16
8. Itel S31 (1GB RAM , 16GB ROM)	5592	724	66	125	142	49	71	3

9. Itel S12 (1GB RAM , 8GB ROM)	2448	345	35	6	46	16	173	23
10. Itel S32 (1GB RAM , 16GB ROM)	7342	954	41	21	87	6	36	12
11. Infinix Zero 4 (3GB RAM,32GB ROM)	1829	902	86	135	322	34	85	1
12. Infinix S2 Pro (3GB RAM,32GB ROM)	1355	302	8	105	83	20	145	22
13. Infinix Zero 5 (6GB RAM, 64GB ROM)	5845	969	65	138	300	40	122	10
14. Infinix Zero 3 (3GB RAM,32GB ROM)	8845	803	35	14	109	16	82	32
15. Infinix Hot 2 (2GB RAM,16GB ROM)	6076	701	86	145	111	33	135	40
16. Techno W5 (2GB RAM,32GB ROM)	7369	312	44	25	240	28	57	44
17. Techno L9 (3GB RAM, 16GBROM)	1337	380	20	38	377	16	96	11
18. Techno Camon CX (3GB RAM, 16GB ROM)	6281	121	73	68	27	28	123	17
19. Techno Spark K7 (1GB RAM , 16GB ROM)	8302	102	59	12	148	30	48	25
20. Techno Phantom 8 (6GB RAM , 64BG ROM)	4464	797	1	85	305	37	181	7

Table 3.5 – Data enumeration results from image files after the devices were reset, using Foremost.

No./ID	JPEG	PNG	MP4	MP3	PDF	MS Word	SQLite	Gif
1. Samsung A5 (3GB RAM , 32GB ROM)	1562	49	86	48	84	10	76	19

2. Samsung S7 (4GB RAM , 32GB ROM)	1450	149	43	25	44	1	40	13
3. Samsung A8 (4GB RAM , 64GB ROM)	2231	395	39	26	13	16	76	24
4. Samsung Galaxy Note 4 (3GB RAM,32GB ROM)	2193	121	16	12	72	11	2	20
5. Samsung – Galaxy Note 5 (4GB RAM , 32GB ROM)	3285	361	16	51	14	16	46	21
6. Itel A11 (512MB RAM , 8GB ROM)	1692	156	9	66	23	13	74	12
7. Itel it1 556 plus (1GB RAM , 8GB ROM)	2551	181	49	3	13	7	91	8

8. ITEL S31 (1GB RAM , 16GB ROM)	2711		353	35	60	72	24	32	2
9. ITEL S12 (1GB RAM , 8GB ROM)	1175		166	16	3	22	8	84	12

<p>10. Itel S32 (1GB RAM , 16GB ROM)</p>	<p>3530</p>	 <p>562</p>	<p>19</p>	<p>12</p>	<p>37</p>	<p>3</p>	<p>16</p>	<p>6</p>	
<p>11. Infinix Zero 4 (3GB</p>	<p>877</p>		<p>432</p>	<p>42</p>	<p>64</p>	<p>160</p>	<p>16</p>	<p>42</p>	<p>1</p>

RAM,32GB ROM)									
12. Infinix S2 Pro (3GB RAM,32GB ROM)	651		150	5	53	41	11	68	12
13. Infinix Zero 5 (6GB RAM, 64GB ROM)	2811		468	33	69	144	23	56	5
14. Infinix Zero 3 (3GB RAM,32GB ROM)	4310		390	17	65	55	9	41	17
15. Infinix Hot 2 (2GB RAM,16GB ROM)	3052		366	43	65	49	18	67	21
16. Techno W5 (2GB RAM,32GB ROM)	3681		152	21	12	116	19	26	22
17. Techno L9 (3GB RAM, 16GBROM)	641		182	10	19	183	8	47	5
18. Techno Camon CX (3GB	3015		59	36	33	14	14	63	11

RAM, 16GB ROM									
19. Techno Spark K7 (1GB RAM , 16GB ROM)	3991		52	31	7	72	16	23	12
20. Techno Phantom 8 (6GB RAM , 64BG ROM)	2145		385	1	41	142	18	85	4

### 3.4.5 Analysis of Data Enumeration Results

The purpose of the data extractions described so far in this chapter was to establish how much data can reasonably be expected to be extracted from Android devices from the most popular brands used in Ghana. By examining how much data could be extracted from the devices before and after performing a factory reset on each device, then computing the percentage of files extracted after reset, the study was able to provide numerical ranges for the amount of data expected in a data extraction exercise performed on the listed brands.

Table 3.2 and Table 3.4 provide the baseline of files extracted by Scalpel and Foremost, respectively, before the Android devices were reset. The values in the two tables represent both valid files as well as false positives which were discovered by the file carving tools.

The values do not follow any particular pattern as they are determined by how the owners of the devices used them. The sole purpose of recording these values is to establish the original values against which the percentage of files recovered can be measured.

Table 3.3 and Table 3.5 provide the number of files extracted by Scalpel and Foremost, respectively, after the devices had been reset. Once again, the values do not take into account which of the files are false positives, same as with the values for before the devices

were reset. As can be seen from the values, the number of files extracted after factory reset are all smaller the number that could be extracted before reset, indicating that some of the files were definitely destroyed in the factory reset. However, a significant number of the files that were on the devices before being reset, persisted despite the reset. Finally, Table 3.6 and Table 3.7 below show the percentages of files extracted by Scalpel and Foremost, respectively. For instance, the tables show that Scalpel was able to extract between 45.1% and 48.9% of JPEG files after factory reset on the available Samsung devices. On Intel devices, Scalpel was able to extract between 59.3% and 68.5% of JPEG files after reset. The disparity in the percentages across brands can be attributed to trade practices by the various manufacturers, relating to proprietary software that are used by the manufacturers as well as their choices for eMMC chips.

Table 3.6 – Percentage of files recovered by Scalpel

No./ID	JPEG	PNG	MP4	MP3	PDF	MS Word	SQLite	Gif			
1. Samsung A5 (3GB RAM, 32GB ROM)	47.6	46.9	47.4	45.0	47.2	57.9	35.2	41.9			
2. Samsung S7 (4GB RAM , 32GB ROM)	45.1	49.2	46.2	48.4	54.7	57.1	39.5	41.2			
3. Samsung A8 (4GB	48.9				46.9	41.5	40.4	49.2	51.3	48.3	46.7

RAM , 64GB ROM)								
4. Samsung Galaxy Note 4 (3GB RAM,32G B ROM)	47. 3	49.1	48. 1	43. 3	46. 5	46. 8	48. 8	50. 0
5. Samsung – Galaxy Note 5 (4GB RAM , 32GB ROM)	46. 8	47.1	45. 1	50. 4	48. 5	46. 2	46. 7	48. 5
6. Itel A11 (512MB RAM , 8GB ROM)	59. 3	59.8	57. 1	58. 9	59. 7	60. 0	58. 7	63. 6
7. Itel it1 556 plus (1GB RAM , 8GB ROM)	62. 8	59.1	62. 5	63. 6	58. 8	66. 7	65. 4	66. 7
8. Itel S31 (1GB RAM , 16GB ROM)	65. 3	66.0	66. 7	67. 9	66. 0	69. 2	67. 1	66. 7
9. Itel S12 (1GB RAM , 8GB ROM)	62. 0	63.2	60. 0	62. 0	62. 1	62. 5	63. 6	61. 1
10. Itel S32 (1GB RAM , 16GB ROM)	68. 5	67.3	68. 2	67. 9	68. 4	66. 7	68. 6	69. 2

11. Infinix Zero 4 (3GB RAM,32GB ROM)	56.5	55.0	55.6	55.1	55.2	54.2	54.6	50.0
---------------------------------------	------	------	------	------	------	------	------	------

<p>12. Infinix S2 Pro (3GB RAM,32GB ROM)</p>	<p>53.6</p>	 <p>53.4</p>	<p>54.9</p>	<p>51.4</p>	<p>52.3</p>	<p>52.1</p>	<p>53.0</p>	<p>52.2</p>
<p>13. Infinix Zero 5 (6GB RAM,</p>	<p>52.2</p>	<p>55.0</p>	<p>56.8</p>	<p>57.1</p>	<p>54.2</p>	<p>59.3</p>	<p>53.6</p>	<p>59.1</p>

64GB ROM)								
14. Infinix Zero 3 (3GB RAM,32GB ROM)	57.8	55.1	54.4	54.7	60.0	55.6	56.3	61.5
15. Infinix Hot 2 (2GB RAM,16GB ROM)	57.3	53.8	56.7	59.2	58.2	46.3	56.7	59.1
16. Techno W5 (2GB RAM,32GB ROM)	61.2	65.1	62.5	75.0	61.1	60.0	56.3	69.6
17. Techno L9 (3GB RAM, 16GBROM)	61.8	60.3	62.5	69.6	68.7	62.5	73.5	76.9
18. Techno Camon CX (3GB RAM, 16GB ROM)	62.6	53.1	68.8	68.1	68.7	72.1	69.2	70.6
19. Techno Spark K7 (1GB RAM , 16GB ROM)	72.7	76.3	63.3	66.7	72.3	64.9	68.8	38.7
20. Techno	69.9	66.9	63.8	50.0	65.5	71.4	62.4	65.0

Phantom 8 (6GB RAM , 64GB ROM)									
--------------------------------	--	--	--	--	--	--	--	--	--

Table 3.7 – Percentage of files recovered by Foremost

No./ID	JPEG	PN G	MP4	MP3	PDF	MS Word	SQLite	Gif
1. Samsung A5 (3GB RAM , 32GB ROM)	41.0	45.4	45.7	44.9	42.6	52.6	45.5	44.2
2. Samsung S7 (4GB RAM , 32GB ROM)	41.9	44.3	43.9	42.4	48.9	100.0	37.7	41.9
3. Samsung A8 (4GB RAM , 64GB ROM)	43.6	44.9	44.3	39.4	52.0	42.1	47.8	48.0
4. Samsung Galaxy Note 4 (3GB RAM, 32GB ROM)	40.1	44.8	40.0	42.9	45.3	39.3	40.0	46.5

5. Samsung – Galaxy Note 5 (4GB RAM , 32GB ROM)	41.4	45. 0		43.2	44.0	46. 7	40.0	37.4	48.8
---	------	----------	--	------	------	----------	------	------	------

6. ITEL A11 (512MB RAM, 8GB ROM)	48.1	48.1	 <p data-bbox="508 163 1144 1680">45.0</p>	48.2	47.9	50.0	48.1	41.4	
7. ITEL IT1 556 PLUS (1GB RAM)	48.4	53.1		54.4	37.5	44.8	50.0	49.2	50.0

, 8GB ROM)									
8. Itel S31 (1GB RAM , 16GB ROM)	48.5	48.8	53.0	48.0	50.7	49.0	45.1	66.7	
9. Itel S12 (1GB RAM , 8GB ROM)	48.0	48.1	45.7	50.0	47.8	50.0	48.6	52.2	
10. Itel S32 (1GB RAM , 16GB ROM)	48.1	58.9	46.3	57.1	42.5	50.0	44.4	50.0	
11. Infinix Zero 4 (3GB RAM,32GB ROM)	47.9	47.9	48.8	47.4	49.7	47.1	49.4	100.0	
12. Infinix S2 Pro (3GB RAM,32GB ROM)	48.0	49.7	62.5	50.5	49.4	55.0	46.9	54.5	
13. Infinix Zero 5 (6GB RAM, 64GB ROM)	48.1	48.3	50.8	50.0	48.0	57.5	45.9	50.0	
14. Infinix Zero 3 (3GB	48.7	48.6	48.6	46.3	50.5	56.3	50.0	53.1	

RAM,32GB ROM)									
15. Infinix Hot 2 (2GB RAM,16GB ROM)	50.2	52.2	50.0	44.8	44.1	54.5	49.6	52.5	
16. Techno W5 (2GB RAM,32GB ROM)	50.0	48.7	47.7	48.0	48.3	67.9	45.6	50.0	
17. Techno L9 (3GB RAM, 16GBROM)	47.9	47.9	50.0	50.0	48.5	50.0	49.0	45.5	
18. Techno Camon CX (3GB RAM, 16GB ROM)	48.0	48.8	49.3	48.5	51.9	50.0	51.2	64.7	
19. Techno Spark K7 (1GB RAM , 16GB ROM)	48.1	51.0	52.5	58.3	48.6	53.3	47.9	48.0	
20. Techno Phantom 8 (6GB RAM , 64BG ROM)	48.1	48.3	100.0	48.2	46.6	48.6	47.0	57.1	

Since a major objective of this study is to develop a method for reducing the number of files that can be extracted after factory reset, the minimum percentages of each brand give the best indicator for what the proposed solution should pass in order to be considered a success. Table 3.8 shows the minimum percentages extracted by Scalpel for each brand used in the study, while Table 3.9 shows the minimum percentages by Foremost.

Table 3.8 – Minimum percentages of files extracted by Scalpel for each brand

Brand	JPEG %	PNG %	MP4 %	MP3 %	PDF %	MS Word %	SQLite %	Gif %
Samsung	45.1	46.9	41.5	40.4	46.5	46.2	35.2	41.2
Itel	59.3	59.1	57.1	58.9	58.8	60.0	58.7	61.1
Infinix	52.2	53.4	54.4	51.4	52.3	46.3	53.0	50.0
Techno	61.2	53.1	62.5	50.0	61.1	60.0	56.3	38.7

Tale 3.9 – Minimum percentages of files extracted by Foremost for each brand

Brand	JPEG %	PNG %	MP4 %	MP3 %	PDF %	MS Word %	SQLite %	Gif %
Samsung	40.1	44.3	40.0	39.4	42.6	39.3	37.4	41.9
Itel	48.0	48.1	45.0	37.5	42.5	49.0	44.4	41.4
Infinix	47.9	47.9	48.6	44.8	44.1	47.1	45.9	50.0
Techno	47.9	47.9	47.7	48.0	46.6	48.6	45.6	45.5

### 3.5 HEADLESS BLOCK SWAPPING

#### 3.5.0 Introduction

Because much data persists on Android devices after a factory reset or delete operation, the study set out to propose a solution to the data persistence problem. The fundamental considerations for the proposed solution was for it to be unobtrusive while quickly recognizing data on the physical disk which has been logically deleted but persists on the device.

The proposed solution, dubbed “Headless Block Swapping” or “HBS”, was conceived as a background process which monitors the file tree to listen for deletions and erase the persistent data from the disk. The proposed solution would run at regular intervals while the device is powered on and perform data sanitization while the operating system is running.

#### 3.5.1 Overview of the Proposed Solution

The proposed solution is a background activity that runs as a kernel module and ensures

that data recovery and file carving tools are unable to recognize files whose data persist on the device's storage media although they have been deleted already. The proposed solution, dubbed "Headless Block Swapping", works by destroying the header information of the deleted files so that data recovery tools are unable to detect the beginning points of the deleted files on the storage media. Since the file carving tools work by scanning for file headers then read the data that follows the header, the Headless Block Swapping procedure effectively cripples the activities of the data recovery tools.

### **3.5.2 Headless Block Swapping (HBS) Conceptual Algorithm**

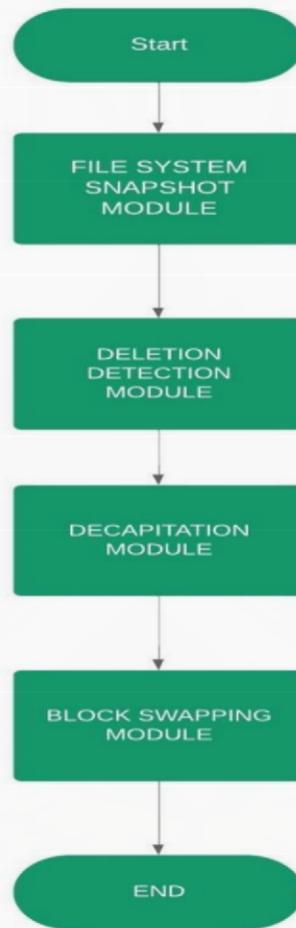
- Take a snapshot of the file system metadata.
- After 10 minutes, take another snapshot of the metadata.
- Compare the two snapshots to identify which files have been deleted since the first snapshot was taken.
- For each deleted file
  - Use the file's inode data from the first snapshot to identify the physical address of the file on the secondary storage medium.
  - Shred the header portion of the file
  - Move blocks of the file to random unallocated blocks on the secondary storage medium.

### **3.5.3 Proposed Components of Headless Block Swapping**

The proposed solution comprises of the following four (4) modules which work sequentially to identify deleted files and destroy traces of them from the Android device:

- File System Snapshot Module (FSSM)

## Headless Block-Swapping



- Deletion Detection Module (DDM)
- Decapitation Module (DM)
- Block Swapping Module (BSM)

Figure 3.25: Headless Block-Swapping

### 3.5.4 File System Snapshot Module

The File System Snapshot Module (FSSM) is the first phase of the proposed solution. In this phase, two major procedures are carried.

- The system recursively traverses the Android file system, enumerating all the files that are in the /sdcard and /data/data directories. That is, the system lists all the files that are explicitly associated with the user's interactions with the Android device.
- For each file that is visited in the traversal, the system extracts its inode data and stores it in a linked list for future processing.

Each time the FSSM runs, a linked list is created containing all the inodes that are linked to active or "undeleted" files at that time, giving a "snapshot" of the file system at the time. When complete, the FSSM triggers the next phase of the Headless Block Swapping process.

```
Def sub fileSystemSnapshot():
```

```
    let initial_dir[] = {'/sdcard', '/data/data'} let
```

```
    inodes = new List
```

```
    for (file in initial_dir):
```

```
        inodes = traverseDir(file,
```

```
        inodes) end for let snapshot = new
```

```
    Snapshot
```

```
    snapshot.setTime(Timestamp.now)
```

```
    snapshot.setInodes(inodes) return
```

```
snapshot end sub
```

```
def sub traverseDir(File file, List inodes):  
    inodes.add(getNode(file)) if  
    (isDirectory(file): let contents =  
    list_contents(file) for (file in  
    contents):  
        inodes = traverseDir(file, inodes)  
end for end if return inodes end sub
```

Listing 3.1: shows an algorithm to carry out the FSSM

### File System Snapshot Module

```
#bin/bash #read  
previous_snapshot=()  
current_snapshot=()  
navi=records  
getPath(){  
    #hold the path to the folder which contains inode files and pass it to next function  
    listFiles  
} listFiles(){  
    #recieve folder path, list all files in folder and find last file    allFiles=(`ls  
$navi`)  
    #get length of files in folder    arr_length=$(( ${#allFiles[*]} ))
```

```

#get the last file(Current)    currentFile_index=${arr_length-1}
#get the last but one (previous)    previousFile_index=${arr_length-
2]    #funciton to read contents of file    readFile
"$navi/${allFiles[$previousFile_index]}" $previousFile_index
readFile "$navi/${allFiles[$currentFile_index]}" $currentFile_index

readFile(){
    #reads last index of previous file line by line else it belong to current snapshot
while read -r line; do    index=${arr_length-2}    if [ $2 -eq $index ]
then    previous_snapshot=("${previous_snapshot[@]}" "$line")
    else
        current_snapshot=("${current_snapshot[@]}" "$line")
    fi
done < "$1"

compareSnapshots(){
#initialize array for deleted paths deletedPaths=()
#compare each file in previous snapshot to current snapshot

for prev in "${previous_snapshot[@]}" do
    del=()    found=0    for current in
"${current_snapshot[@]}" do
}

```

```
}
```

```
#if file is found set found variable to 0 else not found so add to deleted paths
```

```
if [ "$prev" = "$current" ] then found=1 break else
    $prev
```

```
del=$prev
```

```
fi
```

```
done if [ $found -eq 0 ] then
```

```
deletedPaths=("${deletedPaths[@]}" "$del")
```

```
#loop through deleted path and passing whole file information to forprint function
```

```
if [[ "${#deletedPaths[@]}" -eq 0 ]]
```

```
then echo Nothing was deleted
```

```
else for a in
```

```
"${deletedPaths[@]}" do
```

```
forPrint "$a" done
```

```
#zero_override "${deletedPaths[@]}"
```

```
zero_override(){
```

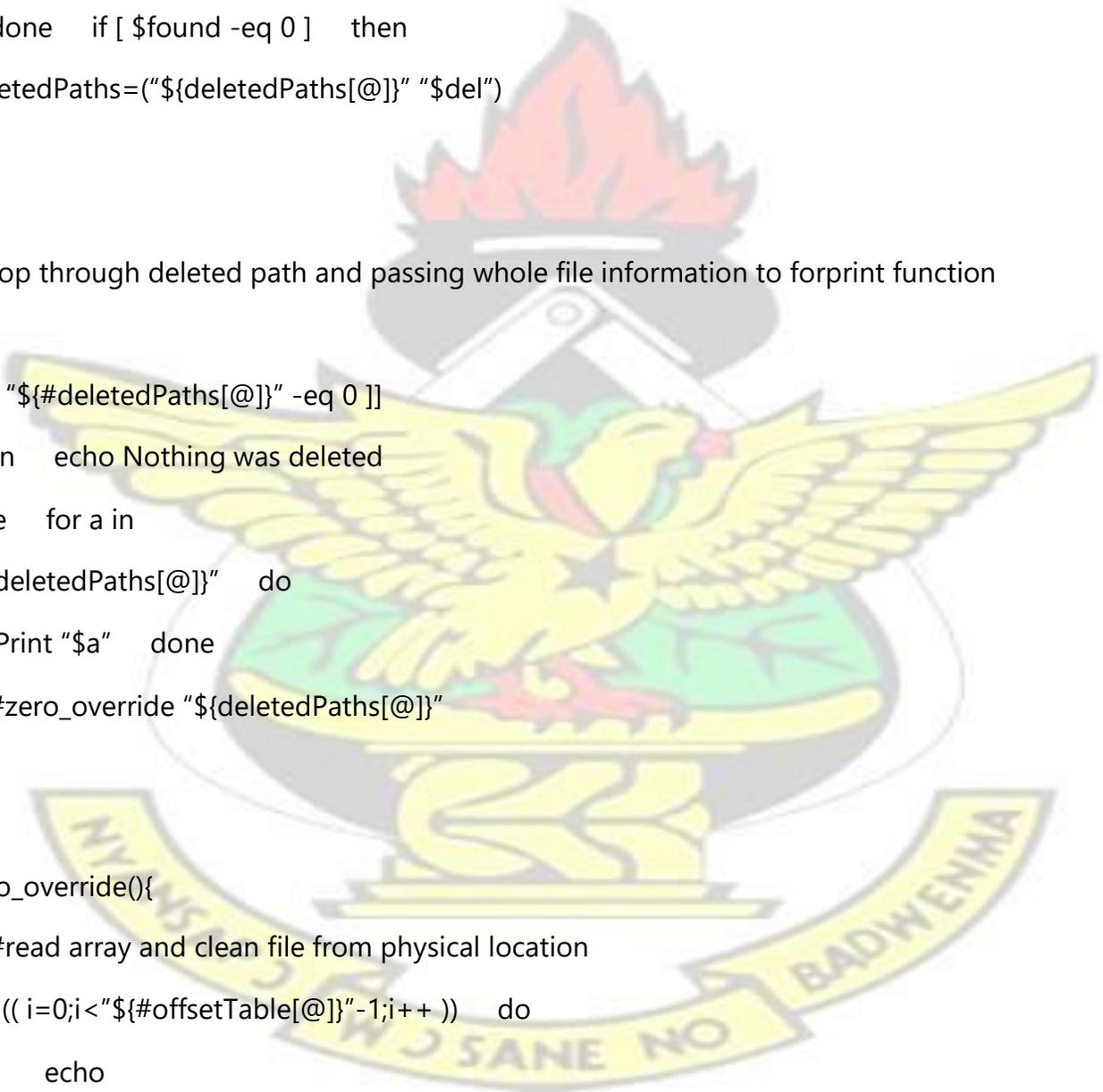
```
#read array and clean file from physical location
```

```
for (( i=0;i<"${#offsetTable[@]}"-1;i++ )) do
```

```
echo
```

```
#echo
```

# KNUST



```
fi  
done
```

```
fi
```

```
}
```

echo The Memory location contains:

```
phys_add="${offsetTable[$i,0]}"
```

```
size=$(( "${offsetTable[$i,1]}" ))    count=$((
size * 4096 ))    input="/dev/zero"
output="/dev/sda"    ibs=1    skip=$((
2048 * 512 + $phys_add * 4096 ))
```

```
check="dd if=$output ibs=$ibs skip=$skip count=$count | hexdump -c"
```

```
override="dd if=$input of=$output seek=$skip count=$count obs=$ibs"
```

```
echo physical address $phys_add
```

```
echo size $count
```

```
echo
```

```
echo "Reading Memory:"
```

```
eval $check
```

```
echo
```

```
echo "Overriding Memory Address"
```

```
eval $override
```

```
echo Done
```

```
echo
```

```
echo checking address again
```

```
eval $check
```

```
#to print data forPrint(){
```

```
done
```

```
}
```

```

#initialising variables declare -A offsetTable
awkseive='{ $1="";$2="";$3="";$4="";$5="";$6="";print}'
arr=("$@") physical_offset=() counterA=0
counterB=0 deletedSize=$(echo $a | awk '{print $4}')
#storing file data in multidimensional array for I in
`echo "${arr[@]}" | awk $awkseive` do if [[ $i = "["
|| $i = "]" || $i = "," ]] then

    elif [[ $i = "," ]]

        ((counterA++))
    counterB=0 else
        offsetTable[$counterA,$counterB]=$i
        ((counterB++))
    fi
done
#just echoing file information ( path and size)
echo echo File information:
echo "${arr[@]}" | awk '{print " \tPath: " $2}'
echo $deletedSize | awk '{print "\tSize: "$0}'

#so if size of array below is 16 it means 8 offsets and 8 lengths and can be referenced
#example of usage "${offsetTable[1,1]}" echo Size of 2D
array: ${#offsetTable[@]} | awk '{print "\t"$0}' zero_override
:

```

then

echo

}

getPath compareSnapshots

# KNUST

Listing 3.2: File System Snapshot Module Pseudocode



## File System Snapshot Module

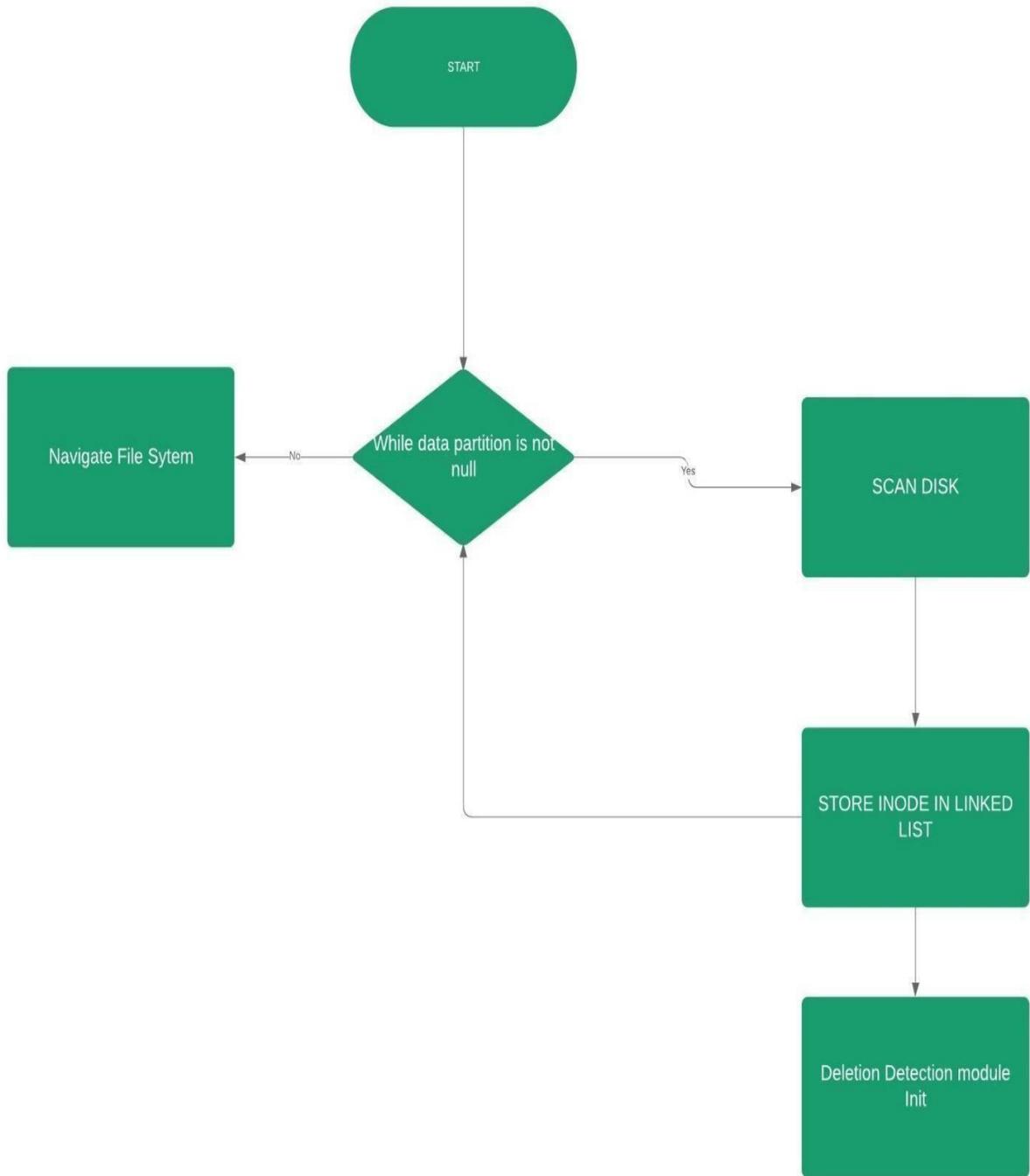


Figure 3.26: File Snapshot Module

### 3.5.2 Deletion Detection Module

The Deletion Detection Module (DDM) is the second phase of the proposed solution. It takes its input from the File System Snapshot Module in the form of snapshots of the inode structure of the file system. The DDM works by comparing two snapshots which were taken one after the other in a specified period of time. The contents of the second snapshot are compared with the contents of the first one in order to identify which of the inodes have been unallocated since the first snapshot was taken. The inodes which are identified as having been unallocated are stored in a linked list for processing in the next phase of the Headless Block Swapping process.

```
Def sub deletionDetection(Snapshot first, Snapshot second):
```

```
    let deleted_inodes = first.getInodes()
```

```
    for (inode in second.getInodes()):
```

```
        deleted_inodes.remove(inode)
```

```
    end for
```

```
    return deleted_inodes
```

```
end for
```

Listing 3.3: Deletion Detection Module

#### Deletion Detection

```
#bin/bash get_data(){
```

```
phys_add=1099900
```

```
dummy="$(dd if=$output ibs=1 skip=$skip count=$count)"
```

```
check_data(){  
    echo  
    echo scanning memory data one data_one=get_data  
    echo  
    echo waiting before next scan  
    sleep 10    echo  
    echo scanning memory data two data_two=get_data  
    echo  
    if [ $data_one = $data_two ]  
then    echo Data persists.  
Delete.  
    #the line below will delete the data from memory address if uncommented  
    #override="$(dd if=$input of=$output seek=$skip count=$count obs=$ibs)"  
    echo fi  
  
check_data
```

Listing 3.4: Deletion Detection

```
skip=$(( 2048 * 512 + $phys_add * 4096 ))  
output="/dev/sda" count=4096
```

}

}

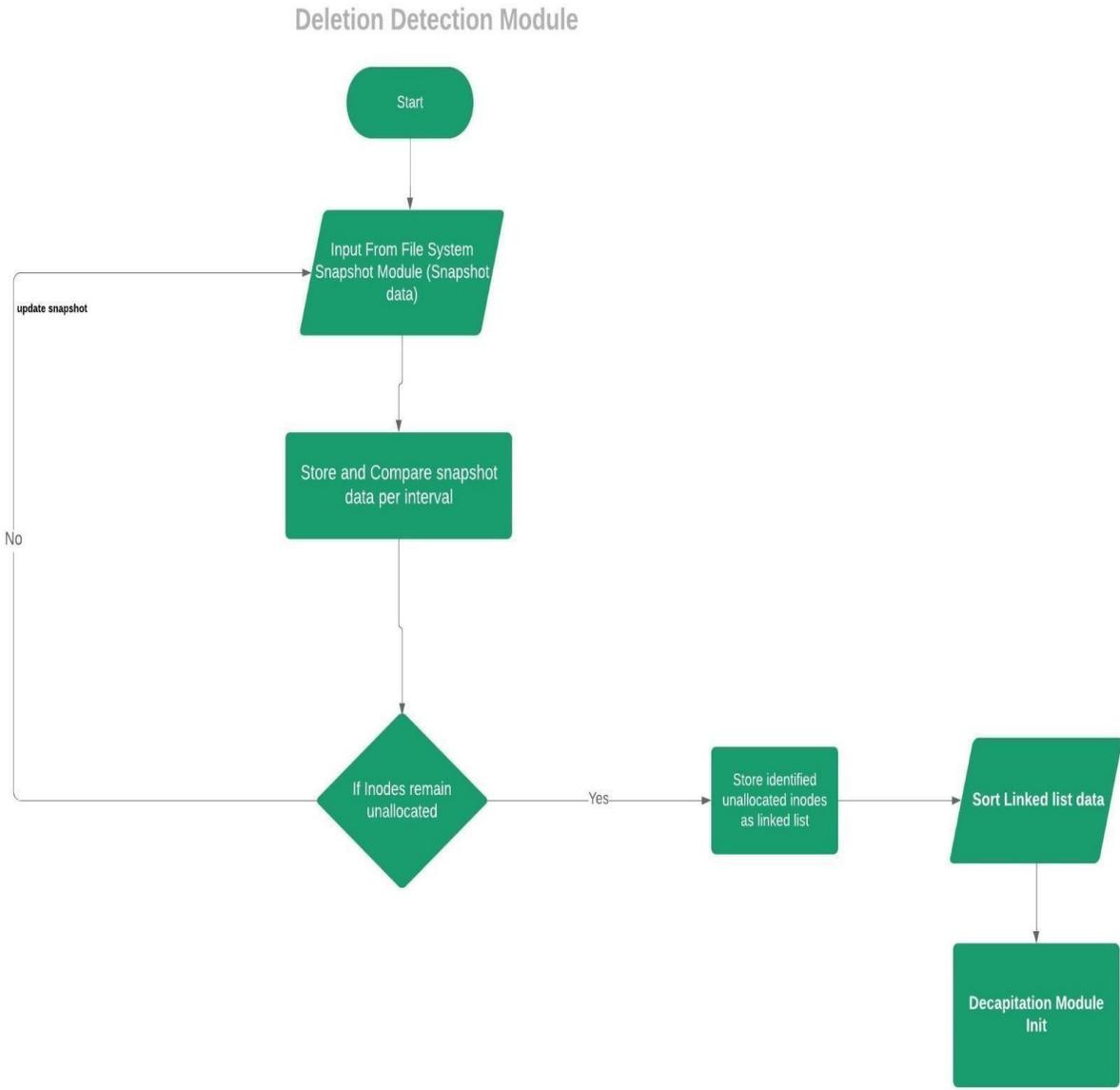


Figure 3.27: Deletion Detection Module

### 3.5.3 Decapitation Module

The Decapitation Module (DM) processes the list of unallocated inodes that was created in the second phase of the Headless Block Swapping process. In this third phase, the

header sections of the “deleted” files are overwritten with NULL values, effectively destroying the headers. This action makes it impossible for data recovery tools that rely on identifying headers and their corresponding footers such as foremost and scalpel, to function effectively (Figure 3.28).

```
def sub decapitation(List deleted_files_inodes):
```

```
    for (inode in deleted_files_inodes):
```

```
        decapitate(inode.getHead())
```

```
    end for
```

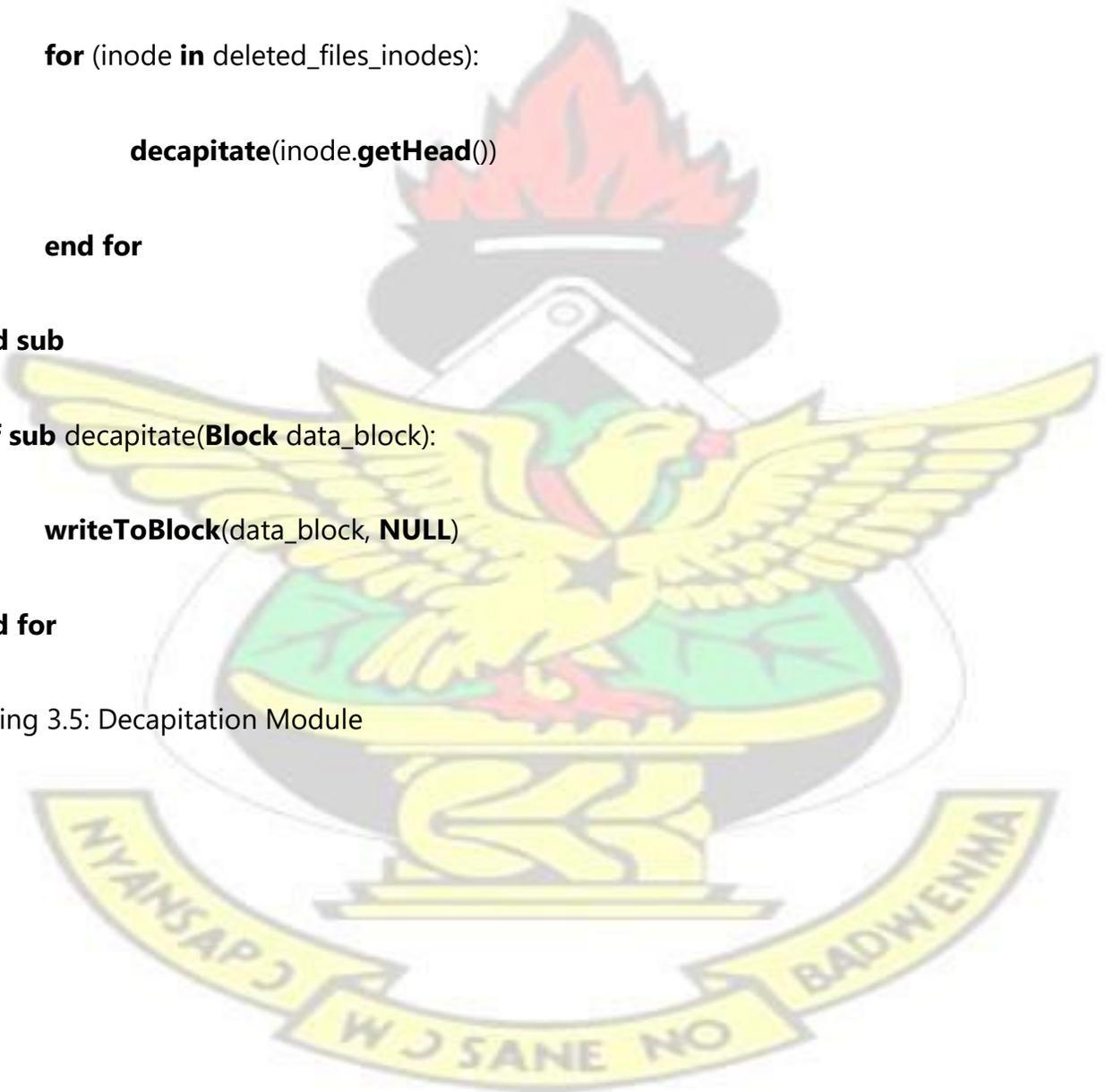
```
end sub
```

```
def sub decapitate(Block data_block):
```

```
    writeToBlock(data_block, NULL)
```

```
end for
```

Listing 3.5: Decapitation Module



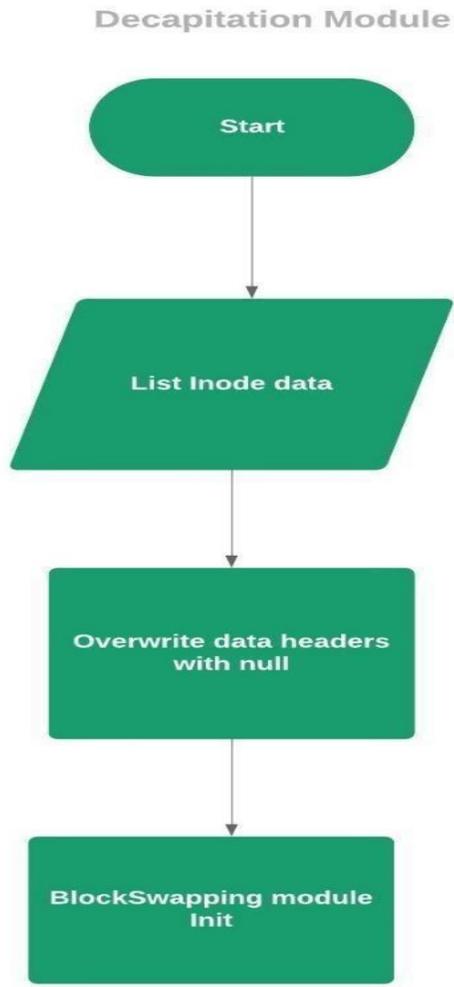
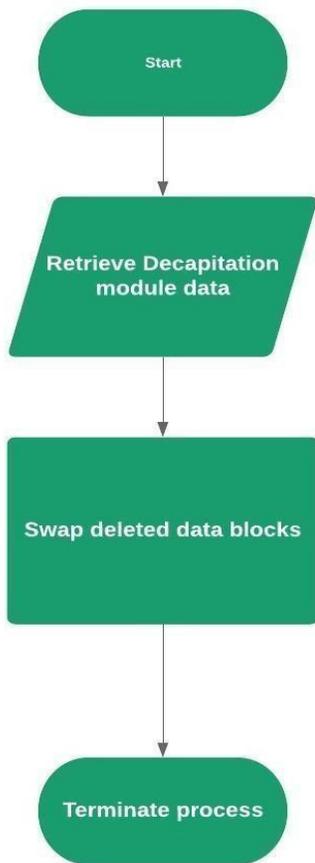


Figure 3.28: Decapitation Module

### **3.5.4 Block Swapping Module**

The last phase of the Headless Block Swapping process makes use of the Block Swapping Module (BSM). This module swaps the contents of the “deleted” file’s data blocks with unallocated space on the disk. By so doing, data recovery tools that detect deleted files by using the encoding pattern of the data on the disk are rendered inoperable (Figure 3.29).

### Block-Swapping Module



### BLOCK-SWAPPING

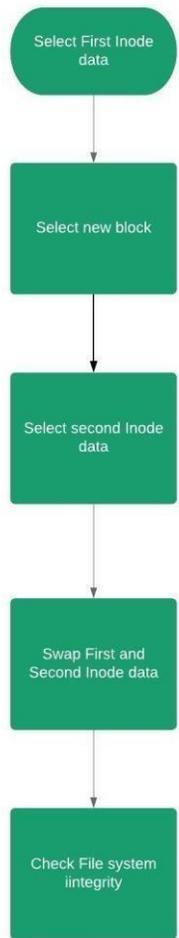


Figure 3.29: Block Swapping Module

```
#bin/bash filePaths=()
thisTime=() table=()
folderName="records"
group(){ getPaths
create_file_with_date
```

```

writing $thisTime } }
#Listing all files in current directory(including sub directories)
getPaths(){ filePaths=(`find *`)
arr_length=$(( ${#filePaths[*]} )
# echo "${#filePaths[@]}"

getData(){ inode=$(stat $1 | grep Inode: | awk '{print $4}') size=$(stat $1 | grep Size:
| awk '{print $2}') physical_offset=$(filefrag -v $1 | awk 'FNR==4' | awk '{print
substr($5, 0, (length($5)))}') physical_off=(`filefrag -v $1| awk 'FNR>3' | awk '{print hi
substr($4, 0, (length($4)-1) }'
| head -n -1)` offset_length=(`filefrag -v $1| awk 'FNR>3' | awk '{print
substr($6,0,(length($6)) }' | head -n -1)`
# echo this is "${offset_length[@]}"
if [ "$physical_offset" = "" ] then
physical_offset="null"

if [ "$size" = "0" ]
then
size="null"

#offset_length_separator "${physical_off[@]}"
127ynchr_length_separator $1 }
#Write to file writing()

fi

```

fi {

#che

ck if

foler

"reco

rds"

exists

if not

creat

e

check

\_fold

er\_exi

sts

```
for each in "${filePaths[@]}" do    getData `pwd`/"$each"    # echo  
"${#table[@]}"    if [ "${#table[@]}" -eq 0 ] then    echo [  
`pwd`/"$each" , $size ] , [ null , null ]>> $folderName/$1    else  
echo [ `pwd`/"$each" , $size ] , "${table[@]}">> $folderName/$1
```

```
table=() done
```

```
#creates file with date as name
```

```
create_file_with_date(){
```

```
thisTime=(`date +%s`)
```

```
check_folder_exists(){
```

```
if [ -d $folderName ]
```

```
else    mkdir  
$folderName
```

```
128ynchr_length_separator(){    for (( i=0;i<="${#physical_off[@]}"-  
1;i++))    do        echo [ "${physical_off[$i]}" , "${offset_length[$i]}" ]  
table=("${table[@]}" [ "${physical_off[$i]}" , "${offset_length[$i]}" ],)
```

```
fi
```

```
}
```

```
}
```

```
then
```

```
:
```

```
fi
```

```
}
```

```
done
```

```
}
```

```
echo echo Taking First Snapshot echo "[
```

Address, Length ]" group echo echo

waiting time before next snapshot...

sleep 10 echo echo Taking

Second Snapshot echo "[

Address, Length ]" group

Listing 3.5: Data Sanitization (Decapitation and Block Swapping) Module

### 3.6 Alternatives to the Headless Block Swapping Process

The study also explores two alternatives to the Headless Block Swapping, namely

- Headless Data Perforation
- Data Wipe Technique

In the cases of the alternatives, the first two phases from the Headless Block Swapping are maintained. That is, both alternatives contain File System Snapshot and Deletion Detection phases, but from there the Data Wipe Technique proceeds to wipe the entire file data by overwriting it with NULL values. On the other hand, the Headless Data Perforation process overwrites some but not all of the data blocks of the deleted file, including the header section.

### 3.7 HBS Simulation

In order to examine the effects of the Headless Block Swapping algorithm on a running Android device, the study developed a shell script to simulate the operation of the HBS and run the script on five Android devices. The HBS shell script is included in full in

Appendix 3 and has also been shown in portion under the various components of the HBS algorithm in the preceding sections. Table 3.10 below shows the Android devices that were used in the HBS simulation.

Table 3.10: Android devices used to test the HBS algorithm

1	Samsung	A5 (3GB RAM, 32GB ROM)	Android 7.0 – Nougat
2	Itel	S32 (1GB RAM, 16GB ROM)	Android 7.0 – Nougat
3	Infinix	Hot 2 (2GB RAM, 16GB ROM)	Android 5.1.1
4	Techno	W5 (2GB RAM, 32GB ROM)	Android 6.0 – Marshmallow
5	Techno	L9 (3GB RAM, 16GB ROM)	Android 6.0 – Marshmallow

The simulation tests followed the steps outlined.

- The Android device was imaged and the file carving tools, Scalpel and Foremost, were used to extract data from the device.
- The HBS shell script was installed to the device and run from the terminal.

- The device was used for one week.
- The device was factory reset.
- Another image of the device memory was taken and the file carving tools were used to extract data from the device again.
- The data extracted before resetting the device was compared with the data extracted after the factory reset.

## CHAPTER FOUR

This chapter presents the analysis data enumerated from devices with HBS running on them and the results and discussions from the survey work.

### 4.1 Analysis of Data Enumeration Results after Applying the Headless Block Swapping Algorithm

In order to determine the effectiveness of Headless Block Swapping algorithm, the files extracted by Scalpel and Foremost before and after factory reset on devices that were running the HBS were taken. Table 4.1 and Table 4.2 show the number of files extracted by Scalpel and Foremost, respectively, before the devices were reset. Then Table 4.3 and table 4.4 show the number of files that were recovered by Scalpel and Foremost after factory reset. The presence of recoverable files on the devices with the HBS algorithm installed can be attributed to operating system files as well as false positives.

Table 4.1: Data Extracted by Scalpel Before Factory Reset

No./ID	JPEG	PNG	MP4	MP3	PDF	MS Word	SQLite	Gif
--------	------	-----	-----	-----	-----	---------	--------	-----

1. Samsung A5 (3GB RAM, 32GB ROM)	1724	597	78	44	340	3	150	34
2. ITEL S32 (1GB RAM, 16GB ROM)	2700	297	20	90	180	17	140	90
3. Infinix Hot 2 (2GB RAM, 16GB ROM)	1900	420	28	140	110	40	177	49
4. Techno W5 (2GB RAM, 32GB ROM)	2400	270	12	3	99	9	18	20
5. Techno L9 (3GB RAM, 16GBROM)	900	125	35	29	330	17	83	14

Table 4.2: Data Extracted by Foremost Before Factory Reset

No./ID	JPEG	PNG	MP4	MP3	PDF	MS Word	Contacts	Gif
1. Samsung A5 (3GB RAM, 32GB ROM)	1808	108	188	107	197	19	167	43
2. ITEL S32 (1GB RAM, 16GB ROM)	942	954	41	21	87	6	36	12
3. Infinix Hot 2 (2GB RAM, 16GB ROM)	2076	701	86	145	111	33	135	40
4. Techno W5 (2GB RAM, 32GB ROM)	1369	312	44	25	240	28	57	44
5. Techno L9 (3GB RAM, 16GBROM)	1337	380	20	38	377	16	96	11

Table 4.3: Data Extracted by Scalpel After Factory Reset

No./ID	JPEG	PNG	MP4	MP3	PDF	MS Word	Contacts	Gif
1. Samsung A5 11 (3GB RAM, 32GB ROM)	11	85	46	41	19	24	56	35
2. ITEL S32 (1GB RAM, 16GB ROM)	33	61	61	100	89	33	69	36
3. Infinix Hot 2 (2GB RAM, 16GB ROM)	46	89	64	99	49	50	46	3
4. Techno W5 (2GB RAM, 32GB ROM)	82	44	36	93	43	30	6	16
5. Techno L9 (3GB RAM, 16GB ROM)	24	55	88	67	20	48	45	10

Table 4.4: Data Extracted by Foremost After Factory Reset

No./ID	JPEG	PNG	MP4	MP3	PDF	MS Word	Contacts	Gif
1. Samsung A5 (3GB RAM, 32GB ROM)	642	32	56	32	59	5	50	12
2. ITEL S32 (1GB RAM, 16GB ROM)	202	286	12	6	26	1	10	3
3. Infinix Hot 2 (2GB RAM, 16GB ROM)	922	210	25	43	33	9	40	12

4. Techno W5 (2GB RAM,32GB ROM)	810	93	13	7	72	8	17	13
5. Techno L9 (3GB RAM, 16GBROM)	401	100	6	11	113	4	28	3

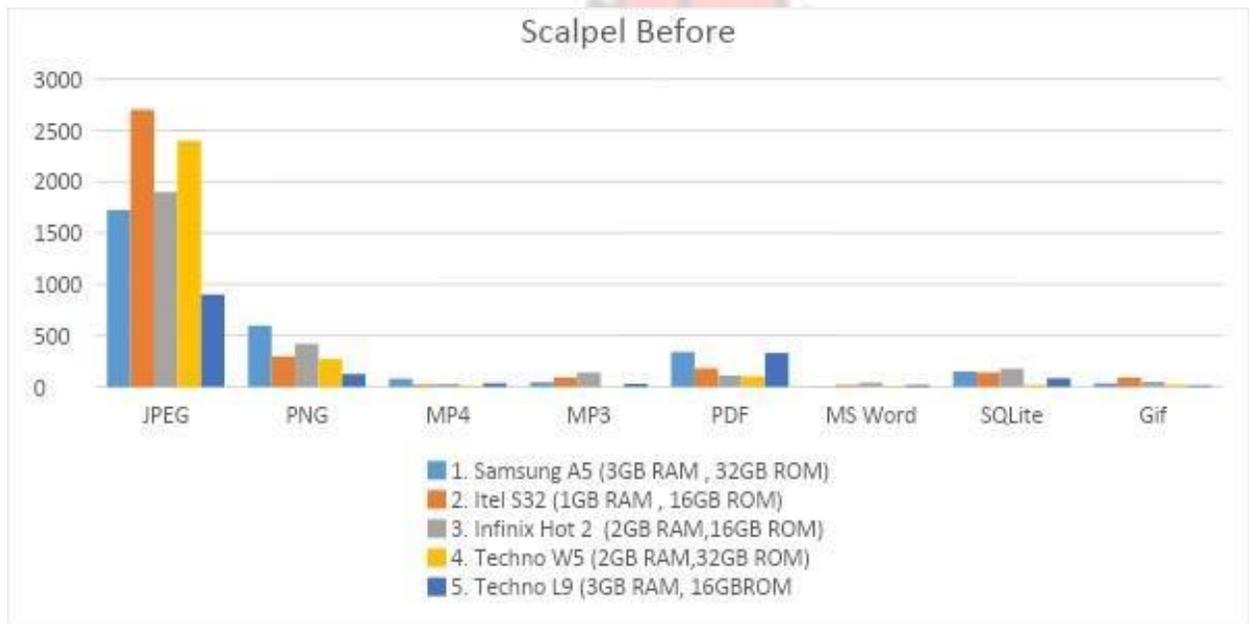
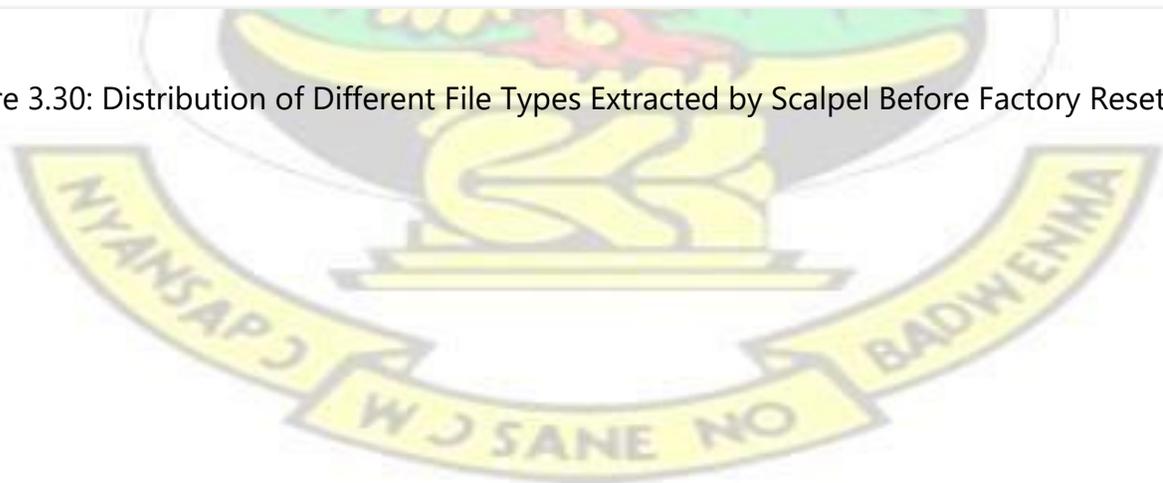


Figure 3.30: Distribution of Different File Types Extracted by Scalpel Before Factory Reset



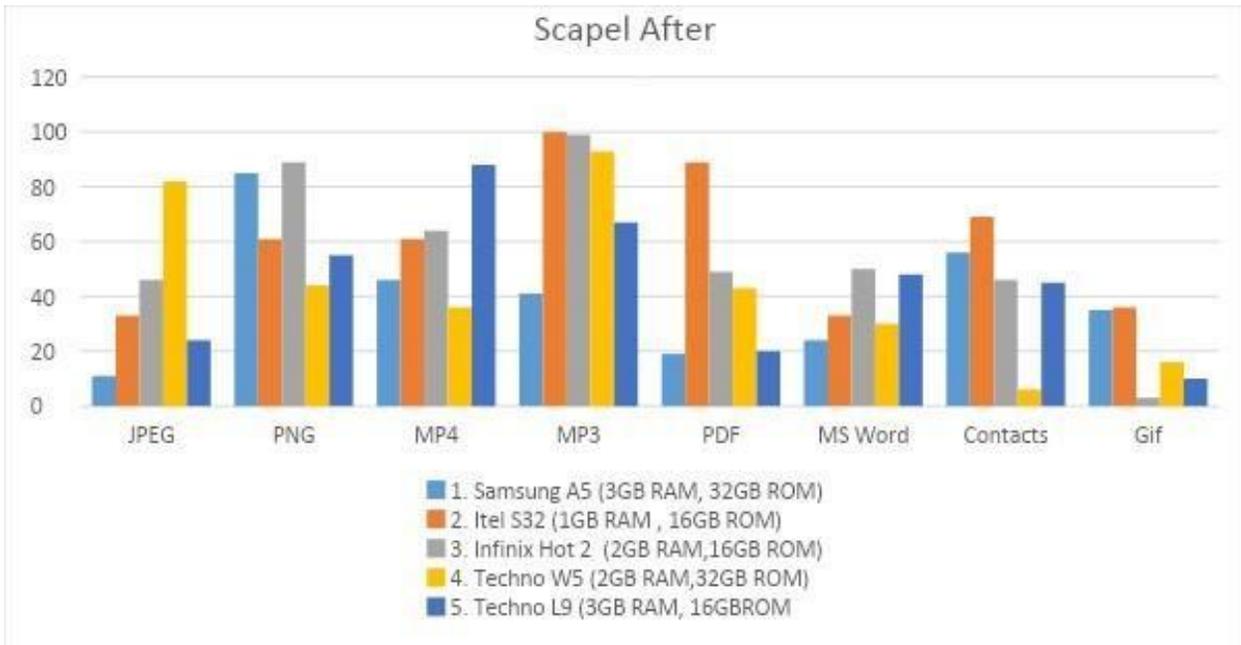


Figure 3.31: Distribution of Different File Types Extracted by Scapel After Factory Reset

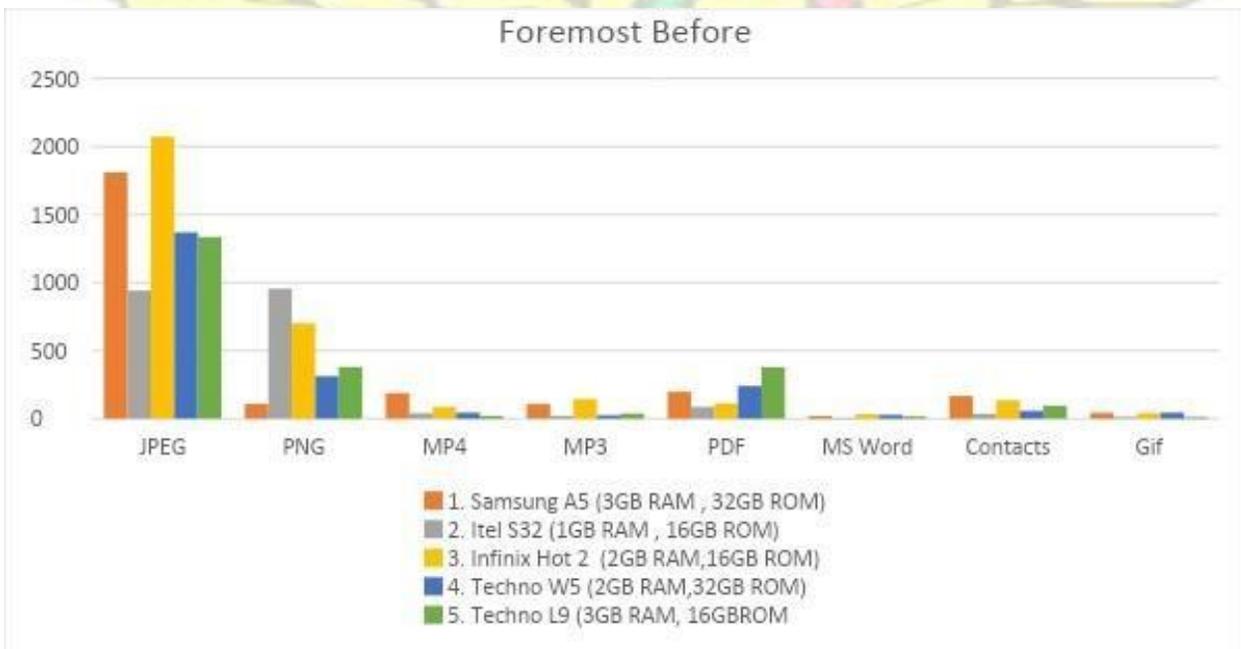


Figure 3.32: Distribution of Different File Types Extracted by Foremost Before Factory Reset

No./ID	JPEG	PNG	MP4	MP3	PDF	MS Word	SQLite	Gif
--------	------	-----	-----	-----	-----	------------	--------	-----

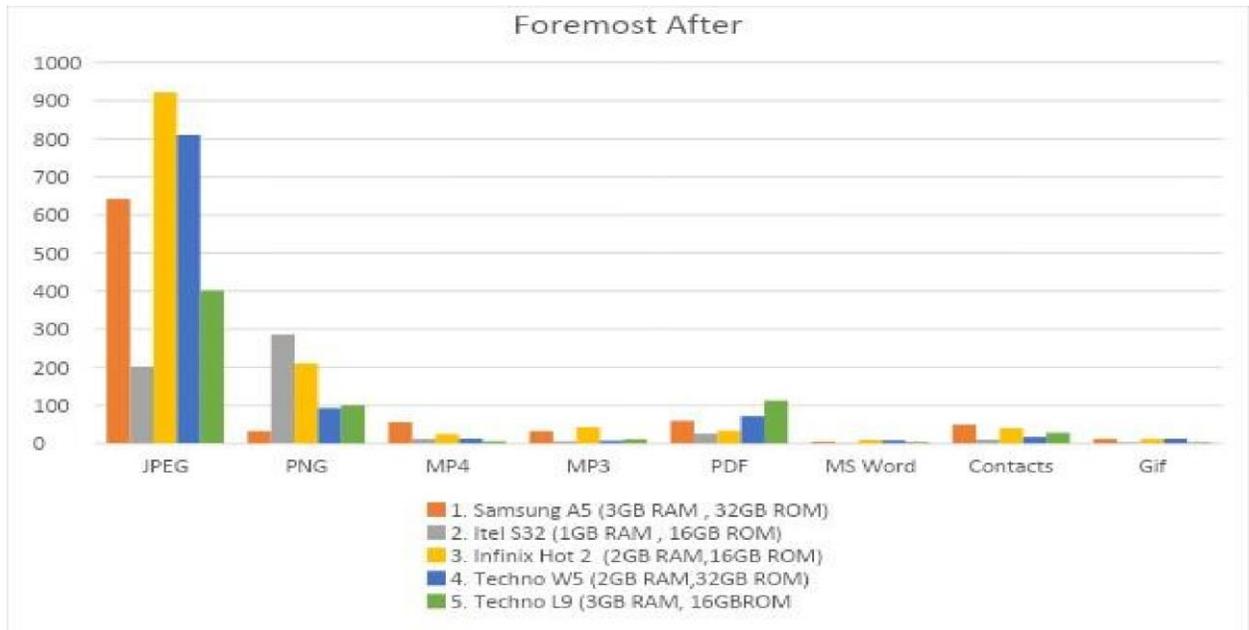


Figure 3.33: Distribution of Different File Types Extracted by Foremost After Factory Reset Finally, Table 4.5 and Table 4.6 show the percentage of files recovered by Scalpel and Foremost.

Table 4.6 – Percentage of files recovered by Scalpel on devices running HBS

No./ID	JPEG	PNG	MP4	MP3	PDF	MS Word	SQLite	Gif
1. Samsung A5 (3GB RAM, 32GB ROM)	19.8	20.4	19.2	20.5	20.9	33.3	18.7	17.6
2. ITEL S32 (1GB RAM, 16GB ROM)	20.1	20.9	25.0	18.9	19.4	23.5	20.7	17.8
3. Infinix Hot 2 (2GB RAM, 16GB ROM)	19.4	18.1	17.9	19.3	21.8	22.5	19.8	20.0
4. Techno W5 (2GB RAM, 32GB ROM)	18.5	23.0	16.7	33.3	21.2	22.2	16.7	25.0
5. Techno L9 (3GB RAM, 16GB ROM)	19.9	20.8	22.9	20.7	19.1	29.4	21.7	28.6

Table 4.7 – Percentage of files recovered by Foremost on devices running HBS

1. Samsung A5 (3GB RAM, 32GB ROM)	30.5	28.7	27.1	34.6	31.0	31.6	28.1	27.9
2. Itel S32 (1GB RAM, 16GB ROM)	30.7	30.5	31.7	38.1	25.3	33.3	30.0	25.0
3. Infinix Hot 2 (2GB RAM,16GB ROM)	30.2	30.4	29.1	28.3	27.9	27.3	32.6	32.5
4. Techno W5 (2GB RAM,32GB ROM)	29.7	29.2	31.8	28.0	29.6	32.1	36.8	34.1
5. Techno L9 (3GB RAM, 16GBROM)	30.1	29.7	30.0	31.6	29.7	31.3	29.2	36.4

Comparing the percentage of files recovered on the devices that had the installation of the HBS script, with the minimum percentages of files recovered on devices without the HBS script, it is readily visible that the recovery rate is far lower for the devices with the proposed solution installed.

## **SURVEY WORK**

### **4.2 Survey Work Results**

The purpose of the study was explained to all the respondents and were assured of the confidentiality and privacy of the information and that, participation was voluntary. Besides this, all the research team presented their identification cards as proof of their affiliation to the Department of Computer Science, Kwame Nkrumah University of Science and Technology (KNUST).

#### **4.2.1 Biodata of Respondents**

From the study result, majority were males (760, 61%), within the 18-24 age cohort (918, 74%), were first degree holders (970, 78%) and were using Android powered smartphones (1040, 84%) as illustrated in Table 4.8.

Table 4.8: Demographic Characteristics of Respondents

Variable	Frequency (N=1240)	Percent (%)
<b>Gender</b>		
Male	760	61.3
Female	480	38.7
<b>Age</b>		
18-24 years	918	74
24-30 years	58	4.7
31 years or more	264	21.3
<b>Educational level</b>		
No education	9	.7
Basic education	27	2.2
SHS/Technical/Vocational	234	18.9
Tertiary	970	78.2
<b>Phone in use</b>		
Android phone	1040	83.9
Windows phone	28	2.3
Apple	114	9.2
Non-smart phone (yam)	58	4.7

#### **4.2.2 Previous phone used and mode of disposal**

The study discovered that almost all respondents across the various demographic category previously owned an Android smartphone. Overall, males (706, 62%) aged 18-24 years (820, 72%) and were first degree holders (880, 77%) had previously used Android phones. However, across all the various demographic spectrum, Samsung smartphones were the main Android phone model used by the respondents. All the responses were significant at  $p < 0.05$  (Table 4.9).

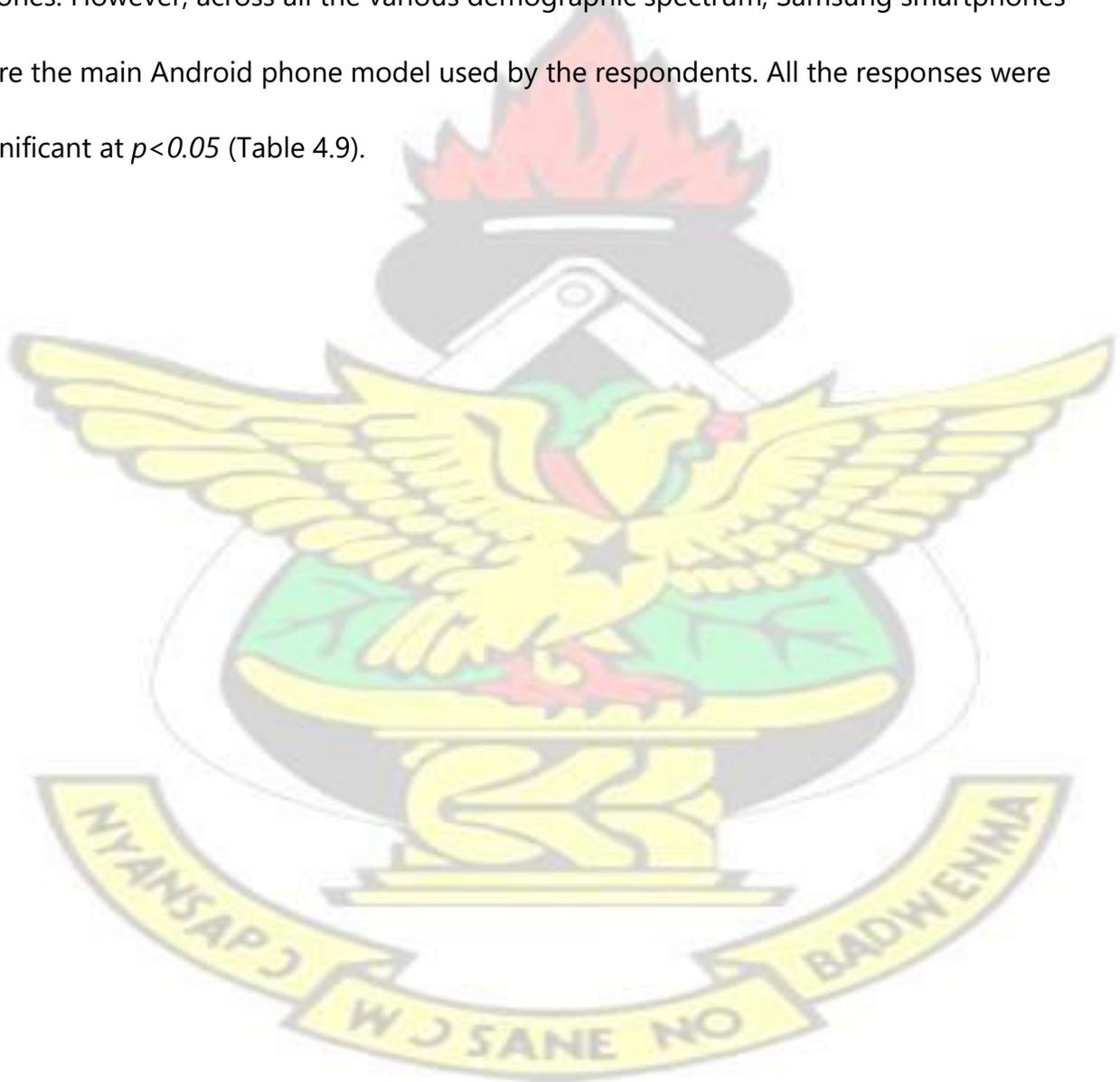


Table 4.9: Chi-square test of previous phone used by biodata of respondents

Variables	Type of phone model used							p-value
	LG	itel	Tecno phones	Infinix or Total	Android Samsung	Other Huawei		
<b>Gender</b>								
Male	358 (62.7)	34 (43)	38 (66.7)	106 (54.4)	91 (69.5)	79 (73.1)	706 (61.9)	0.000*
Female	213 (37.3)	45 (57)	19 (33.3)	89 (45.6)	40 (30.5)	29 (26.8)	435 (38.1)	
<b>Age</b>								
18-24 years	369 (64.6)	49 (62)	47 (82.5)	152 (77.9)	96 (73.3)	107 (99.1)	820 (71.9)	0.000*
24-30 years	41 (7.2)	3 (3.8)	1 (1.7)	7 (3.6)	4 (3.0)	1 (0.9)	57 (5)	
31 years or more	161 (28.2)	27 (34.2)	9 (15.8)	36 (18.5)	31 (23.7)	0 (0)	264 (23.1)	
<b>Educational level</b>								
No education	4 (0.7)	0 (0)	3 (5.3)	0 (0)	2 (1.5)	0 (0)	9 (0.8)	0.000*
Basic education	12 (2.1)	1	0 (0)	8 (4.1)	6 (4.6)	0 (0)	27 (2.4)	
SHS/Technical/ Vocational	107 (18.7)	24	3 (5.3)	33 (16.9)	34 (25.9)	24 (22.2)	225 (19.7)	
Tertiary	448 (78.5)	54	51 (89.5)	154 (79)	89 (67.9)	84 (77.8)	880 (77.1)	

# KNUST

\*The Chi-square statistic is significant at the 0.05 level.

139



Besides the above responses, the study sought information on the mode of disposal of the previous phone depending on whether respondents ever did that. With this, majority were still keeping their previous phones (59%) whereas the remaining had either given it out as a gift, exchanged it, sold it or had the phone stolen (Figure 4.1).

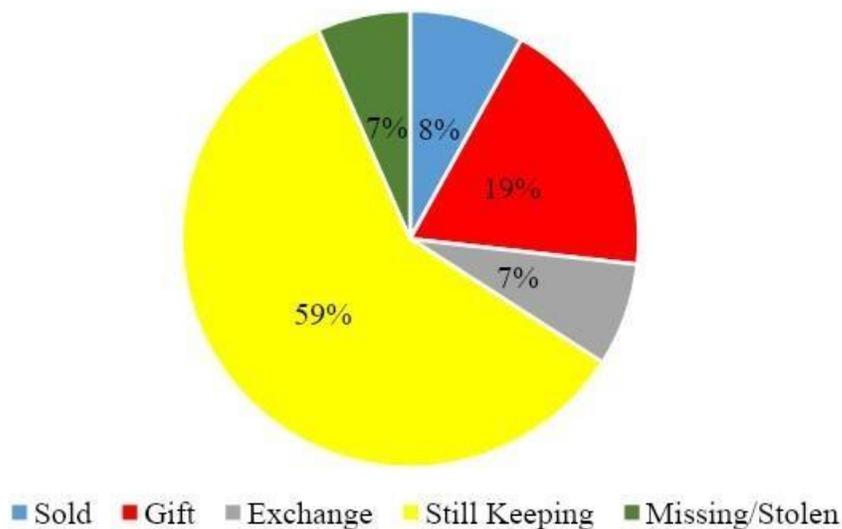


Figure 4.1: Mode of disposal of previous phone

#### 4.2.3 Android phone users, information accessed and data privacy

The study found that 1040 of the study participants used Android smartphones. As regards the type of information frequently accessed on it, these ranged from accessing email, financial transactions, and health, academic and social information (Figure 4.2).

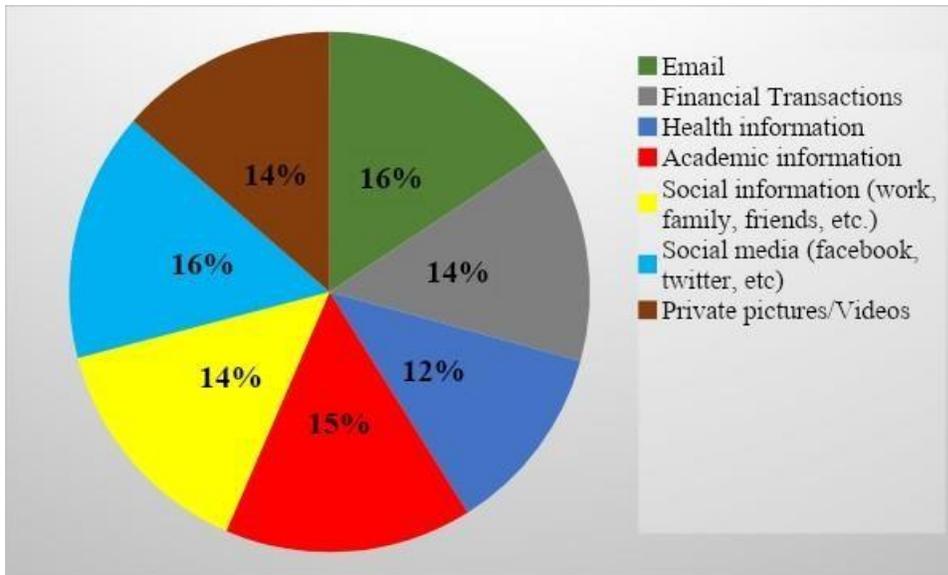


Figure 4.2: Information frequently accessed on phone

Despite this, the type of data considered private or sensitive by respondents include chat history/notes, photos/videos/pictures, office document, contacts and videos (Figure 4.3).

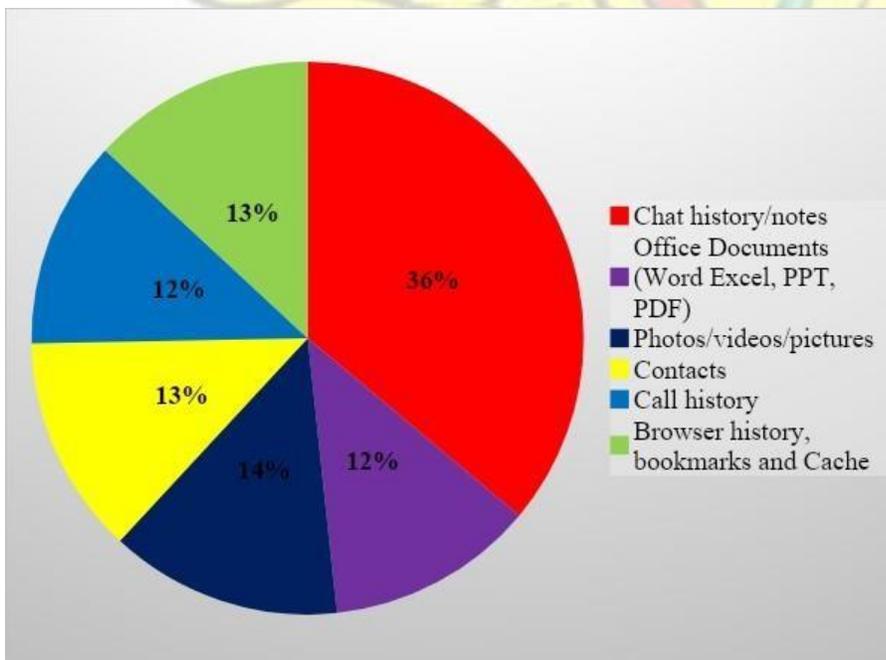


Figure 4.3: Type of data considered private A chi-square test analysis between medium of phone disposal and awareness about

accessing deleted files and means to getting rid of private data before disposing of phone shows that majority (965, 77.8%,  $p=0.094$ ) who either gave out or still kept their phone were aware that data from an Android phone can be accessed after it has been deleted or the device has been reset. This notwithstanding, with regards to the means to getting rid of private data before disposal, whereas majority (500, 40%) of the smartphone users did nothing as regards formatting the memory card (sdcard) ( $p=0.024$ ), some 31% (certain/very certain, 31.5) formatted the memory card. Likewise, whereas 37% did nothing in reference to performing a factory reset ( $p=0.048$ ), some 35% (certain/very certain, 35.3%) undertook this process as illustrated in Table 4.10.

Table 4.10: Chi-square test on medium of phone disposal and private data protection

		Medium of disposing of phone			<i>pvalue</i>
		Gave it out	Still keeping	Total	
		N=496 (%)	N=744 (%)	N=1240 (%)	
Awareness about accessing data after deletion or reset	Yes	374 (75.4)	591 (79.4)	965 (77.8)	0.094
	No	122 (24.6)	153 (20.6)	275 (22.2)	
Delete specific items from the device	Least likely	61 (12.3)	69 (9.3)	130 (10.5)	0.224
	Likely	74 (14.9)	95 (12.8)	169 (13.6)	
	Do Nothing	142 (28.6)	212 (28.5)	354 (28.5)	
	Certain	73 (14.7)	131 (17.6)	204 (16.4)	
	Very certain	146 (29.4)	237 (31.8)	383 (30.9)	

Format the memory card (sdcard)	Least likely	77 (15.5)	118 (15.9)	195 (15.7)	0.024*
------------------------------------	--------------	-----------	------------	---------------	--------

Perform a factory reset on the phone	Likely	79 (15.9)	75 (10.1)	154 (12.4)	0.048*
	Do Nothing	185 (37.3)	315 (42.3)	500 (40.3)	
	Certain	46 (9.3)	82 (11)	128 (10.3)	
	Very certain	109 (22)	154 (20.7)	263 (21.2)	
	Least likely	87 (17.5)	141 (18.9)	228 (18.4)	
Use a "secure deletion" app to erase the phone memory/memory card	Likely	57 (11.5)	58 (7.8)	115 (9.3)	0.446
	Do Nothing	171 (34.5)	288 (38.8)	459 (37)	
	Certain	47 (9.5)	87 (11.7)	134 (10.8)	
	Very certain	134 (27)	170 (22.8)	304 (24.5)	
	Least likely	147 (29.6)	205 (27.5)	352 (28.4)	
Encrypt the phone, then reset	Likely	65 (13.1)	89 (12)	154 (12.4)	0.164
	Do Nothing	220 (44.3)	362 (48.7)	582 (46.9)	
	Certain	34 (6.8)	38 (5.1)	72 (5.8)	
	Very certain	30 (6)	50 (6.7)	80 (6.4)	
	Least likely	145 (29.2)	202 (27.1)	347 (28)	
	Likely	57 (11.5)	61 (8.2)	118 (9.5)	
	Do Nothing	215 (43.3)	367 (49.3)	582 (46.9)	
	Certain	26 (5.2)	42 (5.6)	68 (5.5)	
	Very certain	53 (10.7)	72 (9.7)	125 (10.1)	

square statistic is significant at the 0.05 level.

#### 4.2.4 Knowledge and awareness about Android security and data recovery

Considering the type of information accessed on these phones, as well as those cogitated

as private, the participants' knowledge about the encryption of Android phones were asked using respondents' biodata as predictors. The binary logit model shows that females had a higher odd of knowing about encryption of Android phones [OR=1.609, 95% CI (1.274 – 2.031)] as compared to their male counterparts. The remaining demographic characteristics had some predictive effect but were all insignificant (Table 4.11).

Table 4.11: Predictors of knowledge about encryption of Android phones

Covariates	Do you know of encryption of Android phones? Yes/No		
		OR	95% CI
<b>Gender</b>	Male	1	
	Female	1.609	1.274 – 2.031*
<b>Education</b>	No education	1	
	Basic education	1.101	0.233 – 5.203
	SHS/Technical/Vocational	1.166	0.298 – 4.568
	Tertiary	0.883	0.229 – 3.404
<b>Current used phone</b>	Android phone	1	
	Windows phone	0.702	0.326 – 1.509
	Apple	1.268	0.856 – 1.879
	Non-smart phone (yam)	1.517	0.854 – 2.696

\*Statistically significant at  $p < 0.05$ .

Although majority (78%) of the study participants were knowledgeable about the fact that data from an Android phone can be accessed after it has been deleted or the device has been reset, this had no relation with the demographic characteristics besides their age. The binary logit model in Table 4.12 shows that respondents who were 31 years and above [OR=0.381, 95% CI (0.248-0.585)] had a lesser odd of being aware about Android data recovery in comparison to those between 18 and 24 years.

Table 4.12: Logistic regression model on awareness about Android data recovery

Covariates	Awareness about Android data recovery? Yes/No		
		OR	95% CI
<b>Age</b>	18-24 years	1	
	24-30 years	0.565	0.275-1.160
	31 years and above	0.381	0.248-0.585*
<b>Gender</b>	Male	1	
	Female	1.145	0.867-1.513
<b>Education</b>	No education	1	
	Basic education	0.663	0.127-3.453
	SHS/Technical/Vocational	0.258	0.060-1.117

<b>Current used</b>	Tertiary	0.305	0.072-1.300
	<b>phone</b> Android phone	1	
	Windows phone	0.302	0.070-1.295
	Apple	1.124	0.716-1.765
	Non-smart phone (yam)	0.877	0.411-1.869

\*Statistically significant at  $p < 0.05$ .

Among those who had ever accessed any of their deleted files, the contacts, notes, office documents and pictures were the main files which were retrieved (Figure 4.4).

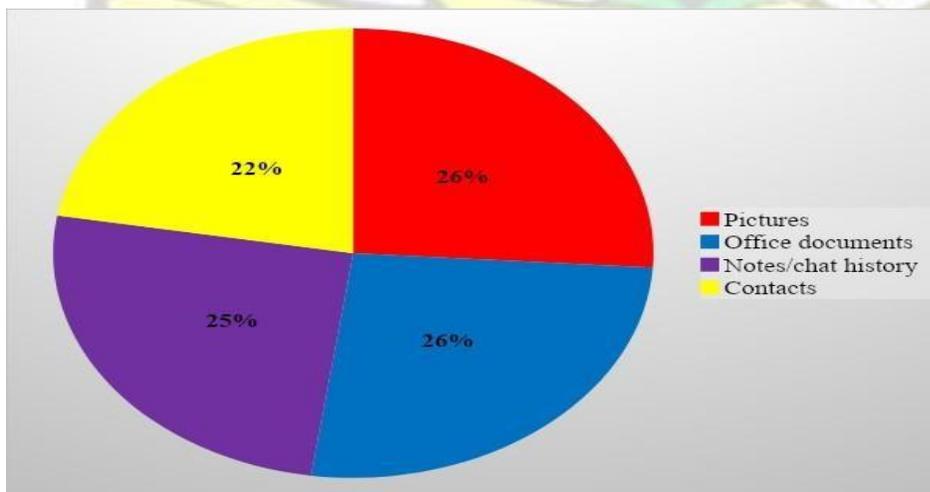


Figure 4.4: Types of files accessed

The study further found that Android apps (17%) and cloud back-up (75%) were the main tools used by respondents to retrieve deleted files on their phones in the event of deletion and/or factory reset as depicted in Figure 4.5.

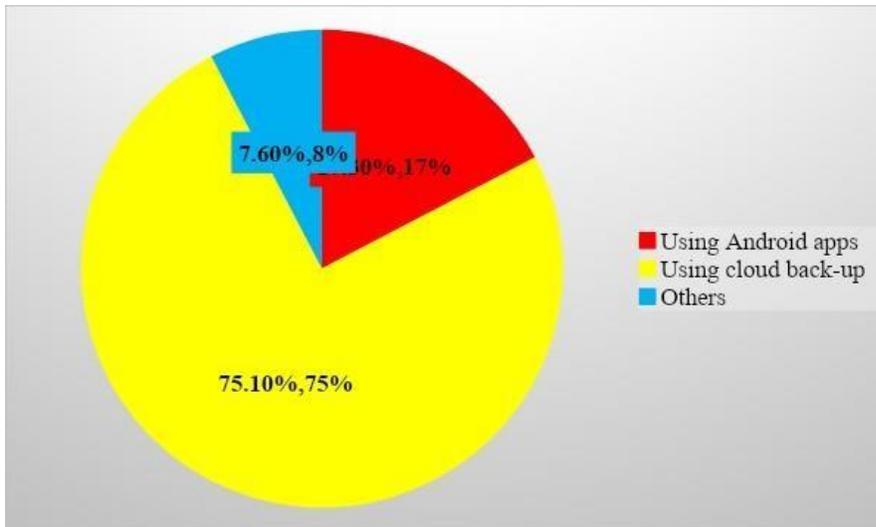


Figure 4.5: Means used to retrieve deleted files

#### 4.2.5 Parameters for securing information on Android phones

In view of the study objective, the data was split by selecting cases of Android smartphone use only among the respondents. Using measures of central tendencies (Mean and standard deviation), the study discovered that whereas an average of 0.2, with a deviation of 0.4 had google account saved on their phone, only an average of 0.04 and a deviation of 0.2 had it 147ynchronized with their phone. Likewise, an average of 0.3 with a deviation of 0.4 had a password or lock pattern on their phone (Table 4.13).

Table 4.13: Parameters for reducing vulnerability of Android smartphones

Parameter	Mean	Standard Deviation
Do you have a google account (gmail account)	0.1606	0.36732
yes, do you log into your account on your phone	0.0378	0.19082

(synchronise)

Was there a password or lock pattern on your phone while in use 0.2779 0.44817

---

What type of phone do you use currently? = Android phone

---

### 4.3 Survey Work Discussion

The current study has assessed the level of vulnerability of Android smartphone users by ascertaining their knowledge and awareness about Android smartphones security in Ghana. Based on the study findings, the predominant activities undertaken on the smartphone include, but not limited to accessing email, financial transactions, and health, academic and social information. This proves that smartphones are effective in accessing personal and corporate email, prepare tax returns, and review customer documents and web browsing, besides its primary function of communication, calling (Dagon, Martin and Starner, 2004; Meshram and Thool, 2014). Although the frequent activities undertaken on the phone included financial transactions and electronic mails, the study participants viewed their photos, videos, audios, office documents, notes and contacts as private information which needs to be protected from the eye of the public. Perhaps, given the upsurge increase in the leak of private and sensitive photos and videos from the smartphones of individuals

with tendencies of destroying their reputation, users tend to place premium on this information. For instance, criminal justice agencies throughout the world are being confronted with an increased need to investigate crimes perpetrated partially or entirely over the internet or other electronic media. Resources and procedures are needed to effectively search for, locate, and preserve all types of electronic evidence. Such evidence according to Lee et al. (2001) range from images of child pornography to encrypted data used to further a variety of criminal activities (Lee, Palmbach and Miller, 2001).

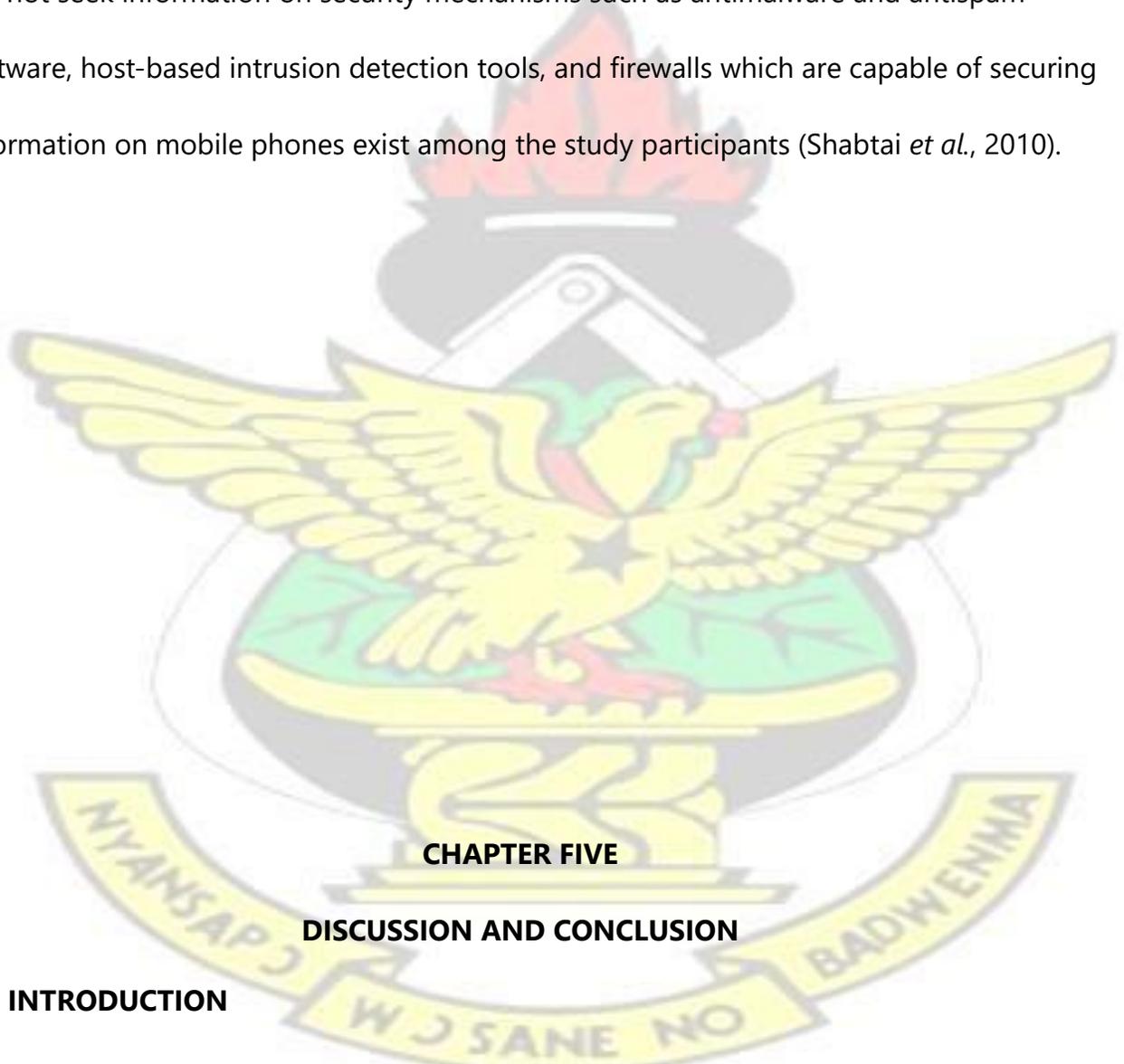
Ironically, though password or lock pattern helps to prevent unauthorized access to one's phone, just a handful of the respondents had their phones protected with either of the two. This behaviour as identified among the study participants is rather unfortunate, given the popularity of Android smartphones, its intrinsic mobility and susceptibility of smartphone data to malicious attacks (Shabtai *et al.*, 2010; Cho and Moon, 2011). This practice rather increases the vulnerability of the device attacks once it is lost, stolen, misused or damaged thus losing sensitive information such as contacts, messages, photos and videos (Ghana News Agency [GNA], 2009). In the Miami phone theft incidence, 54% of the theft victims had no locks, such as passwords or PIN-codes, on their phones hence, increasing the vulnerability of sensitive information to second and third parties (Lost and found, 2011). In our study, considering the fact that a little over half of the respondents (59%) were still keeping their previous Android phones and majority do nothing to retrieve sensitive information in the event of a disposal, such vulnerability compromises the

confidentiality, integrity, and availability of data and services once the phone fall into the hands of malicious people (Dagon, Martin and Starner, 2004; Muslukhov, 2012; Yu *et al.*, 2014). This indicates that personal information stored on smartphones is prone to leakage (Meshram and Thool, 2014).

That notwithstanding, respondents were knowledgeable and aware of the fact that information saved on their smartphones could be retrieved once deleted or in the event of a factory reset. Not only were respondents in the know of it but have ever used it. Contacts, notes, office documents and pictures were the main files ever retrieved by the respondents. In spite of this, the study findings suggest that most of the respondents are not in the know that google backs most of this information for future retrieval once they synchronise their Gmail account on their Smartphones. This is inferred from the abysmal average of 0.04 having 150synchronised their google account though a significant portion of the respondents had googled account. Perhaps, majority of these respondents might not have heard about the on-going universal awareness on the need to secure privacy information or data on Smartphones (Seung-Hyun, Lee and Yim, 2012).

Considering the greater level of security obtained through encryption of smartphones: only persons with the right password can access encrypted data in the event the phone is lost or stolen (Pewter, 2016). The study further reports that respondents' gender significantly predicted their knowledge on encryption of Android phones and age was a

predictor of awareness about Android data security. That notwithstanding, most respondents often do not encrypt their phones though it offers better security in comparison to the usual deletion and factory reset. The limitation to this study is that, it did not seek information on security mechanisms such as antimalware and antispam software, host-based intrusion detection tools, and firewalls which are capable of securing information on mobile phones exist among the study participants (Shabtai *et al.*, 2010).



## **CHAPTER FIVE**

### **DISCUSSION AND CONCLUSION**

#### **5.0 INTRODUCTION**

In this chapter, the findings of the study are summarized and discussed with respect to

the best ways to secure Android devices from data recovery by malicious third parties who have physical access to the device. The chapter talks about the Headless Block Swapping proposed earlier in the study and discusses how it compares against existing data protection solutions.

### **5.1 Assessment of Data Vulnerability**

The study proceeded to quantify the data security performance of the more popular Android device brands and models in Ghana by performing data extraction operations on 20 Android devices from the four (4) brands that are most pervasive in the country, which are Samsung, Itel, Infinix and Techno. The Data Persistence Analysis laid out in Chapter 4 showed that using only Free and Open Source Software, it is possible to obtain images of most Android devices in a relatively short amount of time, further demonstrating the vulnerability of the Android operating system when it comes to sensitive user data.

File enumeration activities conducted during the Data Persistence Analysis phase of the study revealed that Samsung devices have the lowest data persistence, followed by Infinix with a higher data persistence. Techno and Itel devices have the highest data persistence measured in the study. Data persistence in the context of this study did not take into account the existence of false positives but rather did a comprehensive count of files detected by the file carving tools foremost and scalpel. Since the purpose of the study was to examine the rate of file discovery after a factory reset operation, the occurrence of false positives does not come into play in this study. The Data Persistence Analysis compared file discovery for eight (8) different file types including SQLite database files

which are used by the Android operating system to store application data such as contacts, call logs and text messages.

## **5.2 Assessment of Headless Block Swapping**

The proposed solution that was put forward by the study involves monitoring the file system for deletions after which a cleanup operation destroys the data on the physical media. Dubbed, Headless Block Swapping (HBS), the proposed solution was designed to run as a background process with root privileges that continuously works to protect the user from having their data recovered using file carving tools such as foremost and scalpel.

The primary component of the HBS algorithm is the Decapitation Module which is responsible for destroying the header information of the deleted files from the physical media as that is the main way that file carving tools such as foremost and scalpel use to discover files. The second most important component of the HBS, the Block Swapping Module, is responsible for scrambling data from the disk sections where files have been deleted. That second action combats file recovery tools which read data patterns in order to discover and recover files from the physical media.

After implementation of the HBS algorithm, further Data Persistence Analysis of the sample devices showed that data persistence dropped significantly across all devices used in the analysis, confirming the projection made that the HBS algorithm would function effectively to protect

user data on Android devices from third parties who attempt to obtain deleted records from the devices.

Existing alternative solutions for data protection on Android devices involve using a specialized application to delete sensitive files from the device's memory. Such applications require the user to manually indicate which files to delete, from within the application itself. The advantage of an unobtrusive solution like the HBS over the existing alternatives is that it runs in the background without user interference and does not require the user to explicitly indicate which files to destroy. As such the user can perform a delete operation using any application, then the HBS algorithm's Deletion Detection Module is able to recognize the alteration in the file system and initiate the necessary data sanitization processes.

### **5.3 Novelty/Contribution to Knowledge**

The study makes three significant contributions to the academic discussion on the security of Android devices in Ghana and data sanitization in general. Firstly, the study includes data collected on the usage of Android devices specifically in the country of Ghana, which is not readily available in public sources. The data collected and which has been made freely available for academic purposes, is now among the few sources of reliable data on how Ghanaians use mobile devices, especially Android.

Secondly, the study introduces the notion of wiping only the headers of deleted files as

opposed to overwriting the full file body. As previously discussed, the smaller write overhead makes the method described in the study faster and tasks the storage device less. Thirdly, the study contributes the Headless Block Swapping method as an alternative to existing data sanitization methods. The Headless Block Swapping method is independently developed by the author of this study.

#### **5.4 Conclusion**

This study has demonstrated conclusively that Android device security is a real and urgent issue across the world but especially in Ghana. One big aspect of the security discussion is data vulnerabilities presented by file recovery tools of which Free and Open Source Software are well capable of doing the job of extracting data that is intended to be confidential. Among the device brands that are commonly used in Ghana, all major brands show a degree of vulnerability when it comes to file recovery but with the introduction of the Headless Block Swapping algorithm, the vulnerabilities fall to a very low rate.

The study is certain that the way forward for Android data security when it comes to local data is through an unobtrusive data sanitization method such as the Headless Block Swapping algorithm which performs sanitization outside of the user's interference and protects the device

owner from malicious actors who would attack by extracting data from disposed smartphones which have been factory reset or had some sensitive files deleted.

In the case of the survey work, the study generally reports that respondents have a fair knowledge, and were aware about Android phone security. However, this was dependent on their gender and age respectively. Respondents' knowledge centred mainly on data recovery through the use of cloud back-up in the event of deleting sensitive information. Though respondents' knowledge about Android phone encryption was dependent on their gender, they barely used it. Amidst all these, the vulnerability level of respondents to malicious attacks could be rated as high considering the fact that most of the Smartphones were not password protected and participants rarely made any attempt to retrieve information from their phones before disposing it off. Given the sensitivity of Android Smartphones to malicious attacks (Piercy, 2004; Cho and Moon, 2011), the data deletion flaws of Android Smartphones and the increasing infiltration of Android Smartphones into the market of both developed (Gartner Group., 2011) and low- and middle- income countries, especially, Ghana (Mobile Africa, 2015), the study recommends the need for awareness campaign among Smartphone users so as to reduce their tendencies to cybercrime and malicious attacks in the event of theft, misplaced and/or damaged phone. Also, the study recommends an improvement in the Android operating system with regards to data deletion.

# KNUST

## References

- Altuwaijri, H. and Ghouzali, S. (2018) 'Android data storage security: A review', *Journal of King Saud University - Computer and Information Sciences*. The Authors. doi: 10.1016/j.jksuci.2018.07.004.
- Amir, Y. (2016) *Operating Systems 600.418 The File System*. Department of Computer Science, John Hopkins University. Available at: Retrieved July 31, 2016.
- Aouad, L. M. and Kechadi, T. M. (2011) 'Android Forensics a Physical Approach', *Centre of Cybercrime Investigation*.
- Baryamureeba, V. and Florence, T. (2004) 'The Enhanced Digital Investigation Process Model', *Asian Journal of Information Technology*, 5, pp. 790–794.
- Carey, J. (1993) 'Linking qualitative and quantitative methods: Integrating cultural factors into public health.', *Qualitative Health Research.*, 3, pp. 298–318.
- Casey, E. (2011) *Digital Evidence and Computer Crime*, Academic Press, Elsevier Inc.

Chang, Y.-H. and Kuo, T.-W. (2011) 'A reliable MTD design for MLC flash-memory storage systems', p. 179. doi: 10.1145/1879021.1879045.

Cho, T. and Moon, N. (2011) 'Smartphone Application Development and Code-signing', *KIISC*, 21(1), pp. 19–25.

Creswell, J. (2003) *Research design: Qualitative, quantitative and mixed methods approaches*. 2nd ed. Thousand Oaks, CA: SAGE Publications.

Curran, K. et al. (2010) 'Mobile phone forensic analysis', *International Journal of Digital Crime and Forensics*, 2(3), pp. 15–27. doi: 10.4018/jdcf.2010070104.

Dagon, C., Martin, T. and Starner, T. (2004) 'Mobile Phones as Computing Devices: The Viruses Are Coming', *IEEE Pervasive Computing*, 3(4), pp. 11–15.

Delac, G., Silic, M. and Krolo, J. (2011) 'Emerging security threats for mobile platforms', *MIPRO 2011 - 34th International Convention on Information and Communication Technology, Electronics and Microelectronics - Proceedings*. IEEE, pp. 1468–1473.

Dunham, K. (2008) "Mobile Malware Attacks and Defense", Syngress Publishing'.

Etikan, I., Musa, S. and Alkassim, R. (2016) 'Comparison of Convenience Sampling and Purposive Sampling.', *American Journal of Theoretical and Applied Statistics*, 5(1), pp. 1-4.

Flo, A. and Josang, A. (2009) 'Consequences of botnets spreading to mobile devices', *Short-Paper Proceedings of the 14th Nordic Conference on Secure IT Systems (NordSec 2009)*, (October), pp. 37–43. Available at:

<http://persons.unik.no/josang/papers/rfj2009nordsec.pdf>.

Gartner Group. (2011) *Press Release*,.

Gary, L. (2001) 'A Road Map for Digital Forensic Research. Technical Report DTR – T001001, DFRWS. Report for the First Digital', *Forensic Science Communications*, 2(4).

Ghana News Agency [GNA] (2009) *Police: Crime rate in Greater Accra declines in Accra.*

Ghana Statistical Service (GSS) (2013) '2010 Population & Housing Census', *Regional Analytical Report - Upper East Region.*

Guba, E. and Lincoln, Y. (1994) 'Competing paradigms in qualitative research. In: Denzin NK, Lincoln YS, editors.', *Handbook of Qualitative Research. Thousand Oaks, CA;* (105–117).

IBM Corporation (2016) *Component Structure of the Logical File System. IBM Knowledge Center.* Available at:

[https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/File\\_system.html#cite\\_note-IBMKC-7](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/File_system.html#cite_note-IBMKC-7) on March 2018.

International Data Corporation [IDC] (2017) *Smartphone OS Market Share, 2017.*

Joshi, J. and Parekh, C. (2016) 'Android smartphone vulnerabilities: A survey', *Proceedings - 2016 International Conference on Advances in Computing, Communication and Automation, ICACCA 2016.* doi: 10.1109/ICACCA.2016.7578857.

Kim, D. et al. (2012) 'Forensic analysis of android phone using Ext4 file system journal log', *Lecture Notes in Electrical Engineering*, 164 LNEE(VOL. 1), pp. 435–446. doi: 10.1007/978-94-007-4516-2\_44.

Kruse, W. I. and Jay, G. H. (2002) 'Computer Forensics: Incident Response Essentials. Addison – Wesley'.

Lee, H., Palmbach, T. and Miller, M. (2001) *Henry Lee's Crime scene Handbook. London.* 1st editio. Academic Press.

Lee, P. (2012) *Design Research: What is it and Why do it?* Available at:

[www.reboot.org/2012/02/19/design-research-what-is-it-and-why-do-it/](http://www.reboot.org/2012/02/19/design-research-what-is-it-and-why-do-it/) on 25 March 2018.

Lessad, J. and Kessler, G. C. (2013) 'Android forensics: Simplifying cell phone examinations', *Small Scale Digital Device Forensics Journal*, 4(1), pp. 1–12.

Li, Z., Xi, B. and Wu, S. (2016) 'Digital forensics and analysis for Android devices', *ICCSE 2016 - 11th International Conference on Computer Science and Education*, (Iccse), pp. 496–500. doi: 10.1109/ICCSE.2016.7581630.

Lost and found (2011) *The challenges of finding your lost or stolen phone*.

Meshram, P. D. and Thool, R. C. (2014) 'A Survey Paper on Vulnerabilities in Android OS and Security of Android Devices', *Global Conference on Wireless Computing and Networking (GCWCN) IEEE*.

Michael, N., Mark, M. and Lawrence, P. (2000) 'Recovering and Examining Computer Forensic Evidence', *Forensics Science Communications*, 2(4).

Milne, J. (1998) 'Questionnaires: Some Advantages and Disadvantages', *Evaluation cookbook*, p. 88. Available at: <http://www.icbl.hw.ac.uk/ltidi/cookbook/cookbook.pdf>.

Mobile Africa (2015) *Study reveals Ghana mobile phone usage stats*.

Muslukhov, I. (2012) *Survey: Data Protection in Smartphones Against Physical Threats*.

Nitya (2017) 'Content Providers', Available at: [https://github.com/gdg-hudsonvalley/android-guide/tree/master/lifecycle\\_&\\_databases](https://github.com/gdg-hudsonvalley/android-guide/tree/master/lifecycle_&_databases) on March 18 2018.

Noblett, M., Pollitt, M. M. and Presley, L. (2000) 'Recovering and Examining Computer Forensic Evidence', *Forensic Science Communications*, 2(4).

OWASP (2017) 'OWASP Top 10 2017 - The Ten Most Critical Web Application Security

Risks Release Candidate 2', pp. 1–25. Available at:

[https://www.owasp.org/images/b/b0/OWASP\\_Top\\_10\\_2017\\_RC2\\_Final.pdf](https://www.owasp.org/images/b/b0/OWASP_Top_10_2017_RC2_Final.pdf).

Palinkas, L. A. *et al.* (2015) 'Purposeful Sampling for Qualitative Data Collection and Analysis in Mixed Method Implementation Research.', *Adm Policy Ment Health*, 42(5), pp. 533–544.

Palmer, G. L. (2002) 'Forensic Analysis in the Digital World', *International Journal of Digital Evidence*, 1(1), pp. 1–6.

Palmer, J. (2017) *Google says that there are now over 2 billion active Android devices in the world*. Available at: <https://www.androidpolice.com/2017/05/17/google-says-now-2billion-android-devices-world/> on 25 March 2018.

Patton, M. (2002) *Qualitative research and evaluation methods*. 3rd ed. Thousand Oaks, CA: Sage.

Pewter, R. (2016) *The Pros And Cons Of Android Encryption*.

Piercy, M. (2004) 'Embedded devices next on the virus target list.', *IEE Electronics Systems and Software*, 2, pp. 42–43.

Pomeranz, H. (2010) *Understanding ext4*.

Prabhakaran, V. *et al.* (2005) 'IRON file systems', *Proceedings of the twentieth ACM symposium on Operating systems principles - SOSP '05*, p. 206. doi: 10.1145/1095810.1095830.

Quarter, S. (2016) 'Quarterly statistical bulletin on communications in ghana', 1(2).

Racic, R., Ma, D. and Chen, H. (2006) 'Exploiting MMSVulnerabilities to Stealthily Exhaust Mobile Phone's Battery Cellular networks', *Integration The Vlsi Journal*.

Raghavan, S. (2013) 'Digital forensic research: current state of the art', *CSI Transactions on ICT*, 1(1), pp. 91–114. doi: 10.1007/s40012-012-0008-7.

Rao, V. V. and Chakravarthy, A. S. N. (2016) 'Forensic analysis of android mobile devices', *2016 International Conference on Recent Advances and Innovations in Engineering, ICRAIE 2016*, pp. 0–5. doi: 10.1109/ICRAIE.2016.7939540.

Reardon, J., Basin, D. and Capkun, S. (2013) 'SoK: Secure data deletion', *Proceedings - IEEE Symposium on Security and Privacy*. IEEE, pp. 301–315. doi: 10.1109/SP.2013.28.

Reith, M., Carr, C. and Gunsch, G. (2002) 'An Examination of Digital Forensic Models', *International Journal of Digital Evidence*, 1(3), pp. 42–53. doi: 10.1109/SADFE.2009.8.

Rogers, M. *et al.* (2006) 'Computer Forensics Field Triage Process Model', *The Journal of Digital Forensics, Security and Law*, 1(2). doi: 10.15394/jdfsl.2006.1004.

Roy, N. R., Khanna, A. K. and Aneja, L. (2017) 'Android phone forensic: Tools and techniques', *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2016*. IEEE, pp. 605–610. doi: 10.1109/CCAA.2016.7813792.

Scrivens, N. and Lin, X. (2017) 'Android digital forensics', pp. 1–10. doi: 10.1145/3063955.3063981.

Seung-Hyun, S., Lee, D. G. and Yim, K. (2012) 'Analysis on Maliciousness for mobile application', *IMIS*, 7, pp. 126–129.

Shabtai, A. *et al.* (2010) 'Google Android', *Security Privacy, IEEE*, 8(April), pp. 35–44. doi: 10.1109/MSP.2010.2.

Shu, J. *et al.* (2017) 'Why Data Deletion Fails? A Study on Deletion Flaws and Data Remanence in Android Systems', *ACM Transactions on Embedded Computing Systems*, 16(2), pp. 1–22. doi: 10.1145/3007211.

Vardeman, S. and Morris, M. (2003) 'Statistics and Ethics.', *Am. Stat.* 10.1198/0003130031072, 57, pp. 21–26.

Vidas, T., Zhang, C. and Christin, N. (2011) 'Toward a general collection methodology for

Android devices', *DFRWS 2011 Annual Conference*. Elsevier Ltd, 8, pp. S14–S24. doi: 10.1016/j.diin.2011.05.003.

Wagner, D. T. *et al.* (2015) 'Device analyzer: A privacy-aware platform to support research on the android ecosystem', *Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2015*, (December), pp. 5–6. doi: 10.1145/2766498.2774992.

Wilcox, J. (2009) *Two Stories of Smartphones Stolen*.

Yadav, S. *et al.* (2017) 'Android Vulnerabilities and Security', *International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, pp. 204–208.

Yu, X. *et al.* (2014) '(2014). Remotely Wiping Sensitive Data on Stolen Smartphones. h', <http://dx.doi.org/10.1145/2590296.2590318>.

Yusoff, Y., Ismail, R. and Hassan, Z. (2011) 'COMMON PHASES OF COMPUTER FORENSICS INVESTIGATION MODELS Yunus', *International Journal of Computer Science & Information Technology (IJCSIT)*, 3(3). doi: 10.1155/2016/3068347.

Altuwaijri, H. and Ghouzali, S. (2018) 'Android data storage security: A review', *Journal of King Saud University - Computer and Information Sciences*. The Authors. doi: 10.1016/j.jksuci.2018.07.004.

Amir, Y. (2016) *Operating Systems 600.418 The File System*. Department of Computer Science, John Hopkins University. Available at: Retrieved July 31, 2016.

Aouad, L. M. and Kechadi, T. M. (2011) 'Android Forensics a Physical Approach', *Centre of Cybercrime Investigation*.

Baryamureeba, V. and Florence, T. (2004) 'The Enhanced Digital Investigation Process Model', *Asian Journal of Information Technology*, 5, pp. 790–794.

- Carey, J. (1993) 'Linking qualitative and quantitative methods: Integrating cultural factors into public health.', *Qualitative Health Research.*, 3, pp. 298–318.
- Casey, E. (2011) *Digital Evidence and Computer Crime*, Academic Press, Elsevier Inc.
- Chang, Y.-H. and Kuo, T.-W. (2011) 'A reliable MTD design for MLC flash-memory storage systems', p. 179. doi: 10.1145/1879021.1879045.
- Cho, T. and Moon, N. (2011) 'Smartphone Application Development and Code-signing', *KIISC*, 21(1), pp. 19–25.
- Creswell, J. (2003) *Research design: Qualitative, quantitative and mixed methods approaches*. 2nd ed. Thousand Oaks, CA: SAGE Publications.
- Curran, K. et al. (2010) 'Mobile phone forensic analysis', *International Journal of Digital Crime and Forensics*, 2(3), pp. 15–27. doi: 10.4018/jdcf.2010070104.
- Dagon, C., Martin, T. and Starner, T. (2004) 'Mobile Phones as Computing Devices: The Viruses Are Coming', *IEEE Pervasive Computing*, 3(4), pp. 11–15.
- Delac, G., Silic, M. and Krolo, J. (2011) 'Emerging security threats for mobile platforms', *MIPRO 2011 - 34th International Convention on Information and Communication Technology, Electronics and Microelectronics - Proceedings*. IEEE, pp. 1468–1473.
- Dunham, K. (2008) "Mobile Malware Attacks and Defense", Syngress Publishing'.
- Etikan, I., Musa, S. and Alkassim, R. (2016) 'Comparison of Convenience Sampling and Purposive Sampling.', *American Journal of Theoretical and Applied Statistics*, 5(1), pp. 1-4.
- Flo, A. and Josang, A. (2009) 'Consequences of botnets spreading to mobile devices', *Short-Paper Proceedings of the 14th Nordic Conference on Secure IT Systems (NordSec 2009)*, (October), pp. 37–43. Available at:  
<http://persons.unik.no/josang/papers/rfj2009nordsec.pdf>.

Gartner Group. (2011) *Press Release*.

Gary, L. (2001) 'A Road Map for Digital Forensic Research. Technical Report DTR – T001001, DFRWS. Report for the First Digital', *Forensic Science Communications*, 2(4).

Ghana News Agency [GNA] (2009) *Police: Crime rate in Greater Accra declines in Accra*.

Ghana Statistical Service (GSS) (2013) '2010 Population & Housing Census', *Regional Analytical Report - Upper East Region*.

Guba, E. and Lincoln, Y. (1994) 'Competing paradigms in qualitative research. In: Denzin NK, Lincoln YS, editors.', *Handbook of Qualitative Research*. Thousand Oaks, CA; (105–117).

IBM Corporation (2016) *Component Structure of the Logical File System*. IBM Knowledge Center. Available at: [https://ipfs.io/ipfs/QmXoyvizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/File\\_system.html#cite\\_note-IBMKC-7](https://ipfs.io/ipfs/QmXoyvizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/File_system.html#cite_note-IBMKC-7) on March 2018.

International Data Corporation [IDC] (2017) *Smartphone OS Market Share, 2017*.

Joshi, J. and Parekh, C. (2016) 'Android smartphone vulnerabilities: A survey', *Proceedings - 2016 International Conference on Advances in Computing, Communication and Automation, ICACCA 2016*. doi: 10.1109/ICACCA.2016.7578857.

Kim, D. et al. (2012) 'Forensic analysis of android phone using Ext4 file system journal log', *Lecture Notes in Electrical Engineering*, 164 LNEE(VOL. 1), pp. 435–446. doi: 10.1007/978-94-007-4516-2\_44.

Kruse, W. I. and Jay, G. H. (2002) 'Computer Forensics: Incident Response Essentials. Addison – Wesley'.

Lee, H., Palmbach, T. and Miller, M. (2001) *Henry Lee's Crime scene Handbook*. London. 1st editio. Academic Press.

- Lee, P. (2012) *Design Research: What is it and Why do it?* Available at: [www.reboot.org/2012/02/19/design-research-what-is-it-and-why-do-it/](http://www.reboot.org/2012/02/19/design-research-what-is-it-and-why-do-it/) on 25 March 2018.
- Lessad, J. and Kessler, G. C. (2013) 'Android forensics: Simplifying cell phone examinations', *Small Scale Digital Device Forensics Journal*, 4(1), pp. 1–12.
- Li, Z., Xi, B. and Wu, S. (2016) 'Digital forensics and analysis for Android devices', *ICCSE 2016 - 11th International Conference on Computer Science and Education*, (Iccse), pp. 496–500. doi: 10.1109/ICCSE.2016.7581630.
- Lost and found (2011) *The challenges of finding your lost or stolen phone*.
- Meshram, P. D. and Thool, R. C. (2014) 'A Survey Paper on Vulnerabilities in Android OS and Security of Android Devices', *Global Conference on Wireless Computing and Networking (GCWCN) IEEE*.
- Michael, N., Mark, M. and Lawrence, P. (2000) 'Recovering and Examining Computer Forensic Evidence', *Forensics Science Communications*, 2(4).
- Milne, J. (1998) 'Questionnaires: Some Advantages and Disadvantages', *Evaluation cookbook*, p. 88. Available at: <http://www.icbl.hw.ac.uk/ltidi/cookbook/cookbook.pdf>.
- Mobile Africa (2015) *Study reveals Ghana mobile phone usage stats*.
- Muslukhov, I. (2012) *Survey: Data Protection in Smartphones Against Physical Threats*.
- Nitya (2017) 'Content Providers'. Available at: <https://github.com/gdg-hudsonvalley/android->

guide/tree/master/lifecycle\_&\_databases on March 18 2018.

Noblett, M., Pollitt, M. M. and Presley, L. (2000) 'Recovering and Examining Computer Forensic Evidence', *Forensic Science Communications*, 2(4).

OWASP (2017) 'OWASP Top 10 2017 - The Ten Most Critical Web Application Security Release Candidate 2', pp. 1–25. Available [https://www.owasp.org/images/b/b0/OWASP\\_Top\\_10\\_2017\\_RC2\\_Final.pdf](https://www.owasp.org/images/b/b0/OWASP_Top_10_2017_RC2_Final.pdf).

Palinkas, L. A. *et al.* (2015) 'Purposeful Sampling for Qualitative Data Collection and Analysis in Mixed Method Implementation Research.', *Adm Policy Ment Health*, 42(5), pp. 533–544.

Palmer, G. L. (2002) 'Forensic Analysis in the Digital World', *International Journal of Digital Evidence*, 1(1), pp. 1–6.

Palmer, J. (2017) *Google says that there are now over 2 billion active Android devices in the world.* Available at: <https://www.androidpolice.com/2017/05/17/google-says-now-2billion-android-devices-world/> on 25 March 2018.

Patton, M. (2002) *Qualitative research and evaluation methods*. 3rd ed. Thousand Oaks, CA:

Pewter, R. (2016) *The Pros And Cons Of Android Encryption*.

Piercy, M. (2004) 'Embedded devices next on the virus target list.', *IEE Electronics Systems and Software*, 2, pp. 42–43.

Pomeranz, H. (2010) *Understanding ext4*.

Prabhakaran, V. *et al.* (2005) 'IRON file systems', *Proceedings of the twentieth ACM Sage*.

*symposium on Operating systems principles - SOSp '05*, p. 206. doi:

10.1145/1095810.1095830.

Quarter, S. (2016) 'Quarterly statistical bulletin on communications in Ghana', 1(2).

Racic, R., Ma, D. and Chen, H. (2006) 'Exploiting MMS Vulnerabilities to Stealthily Exhaust Mobile Phone's Battery Cellular networks', *Integration The VLSI Journal*.

Raghavan, S. (2013) 'Digital forensic research: current state of the art', *CSI Transactions on ICT*, 1(1), pp. 91–114. doi: 10.1007/s40012-012-0008-7.

Rao, V. V. and Chakravarthy, A. S. N. (2016) 'Forensic analysis of android mobile devices', *2016 International Conference on Recent Advances and Innovations in Engineering, ICRAIE 2016*, pp. 0–5. doi: 10.1109/ICRAIE.2016.7939540.

Reardon, J., Basin, D. and Capkun, S. (2013) 'SoK: Secure data deletion', *Proceedings - IEEE Symposium on Security and Privacy*. IEEE, pp. 301–315. doi: 10.1109/SP.2013.28.

Reith, M., Carr, C. and Gunsch, G. (2002) 'An Examination of Digital Forensic Models', *International Journal of Digital Evidence*, 1(3), pp. 42–53. doi: 10.1109/SADFE.2009.8.

Rogers, M. et al. (2006) 'Computer Forensics Field Triage Process Model', *The Journal of Digital Forensics, Security and Law*, 1(2). doi: 10.15394/jdfsl.2006.1004.

Roy, N. R., Khanna, A. K. and Aneja, L. (2017) 'Android phone forensic: Tools and techniques', *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2016*. IEEE, pp. 605–610. doi: 10.1109/CCAA.2016.7813792.

Scrivens, N. and Lin, X. (2017) 'Android digital forensics', pp. 1–10. doi: 10.1145/3063955.3063981.

Seung-Hyun, S., Lee, D. G. and Yim, K. (2012) 'Analysis on Maliciousness for mobile application', *IMIS*, 7, pp. 126–129.

Shabtai, A. *et al.* (2010) 'Google Android';, *Security Privacy, IEEE*, 8(April), pp. 35–44. doi: 10.1109/MSP.2010.2.

Shu, J. *et al.* (2017) 'Why Data Deletion Fails? A Study on Deletion Flaws and Data Remanence in Android Systems', *ACM Transactions on Embedded Computing Systems*, 16(2), pp. 1–22. doi: 10.1145/3007211.

Vardeman, S. and Morris, M. (2003) 'Statistics and Ethics.', *Am. Stat.* 10.1198/0003130031072, 57, pp. 21–26.

Vidas, T., Zhang, C. and Christin, N. (2011) 'Toward a general collection methodology for Android devices', *DFRWS 2011 Annual Conference*. Elsevier Ltd, 8, pp. S14–S24. doi: 10.1016/j.diin.2011.05.003.

Wagner, D. T. *et al.* (2015) 'Device analyzer: A privacy-aware platform to support research on the android ecosystem', *Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2015*, (December), pp. 5–6. doi: 10.1145/2766498.2774992.

Wilcox, J. (2009) *Two Stories of Smartphones Stolen*.

Yadav, S. *et al.* (2017) 'Android Vulnerabilities and Security', *International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, pp. 204–208.

Yu, X. *et al.* (2014) '(2014). Remotely Wiping Sensitive Data on Stolen Smartphones. h', <http://dx.doi.org/10.1145/2590296.2590318>.

Yusoff, Y., Ismail, R. and Hassan, Z. (2011) 'COMMON PHASES OF COMPUTER FORENSICS INVESTIGATION MODELS Yunus', *International Journal of Computer Science & Information Technology (IJCSIT)*, 3(3). doi: 10.1155/2016/3068347.

# KNUST



## Appendices

### Appendix 1



Figure 3.1 - Kingo Root Windows(R) application for rooting Android devices

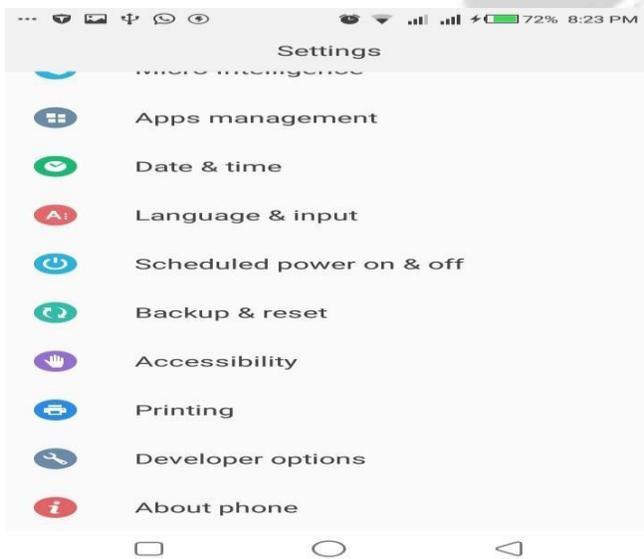


Figure 3.2 – Android Settings app

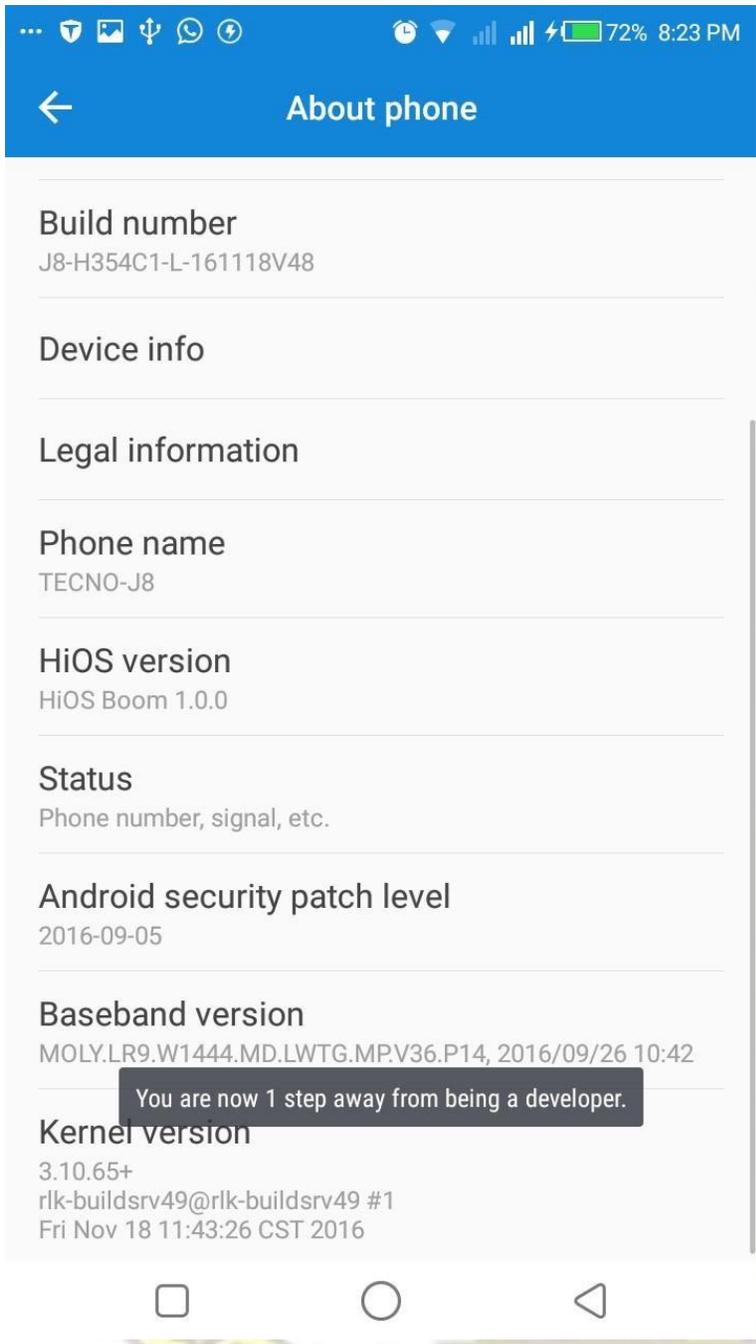


Figure 3.3 – About Phone section of the Android Settings app

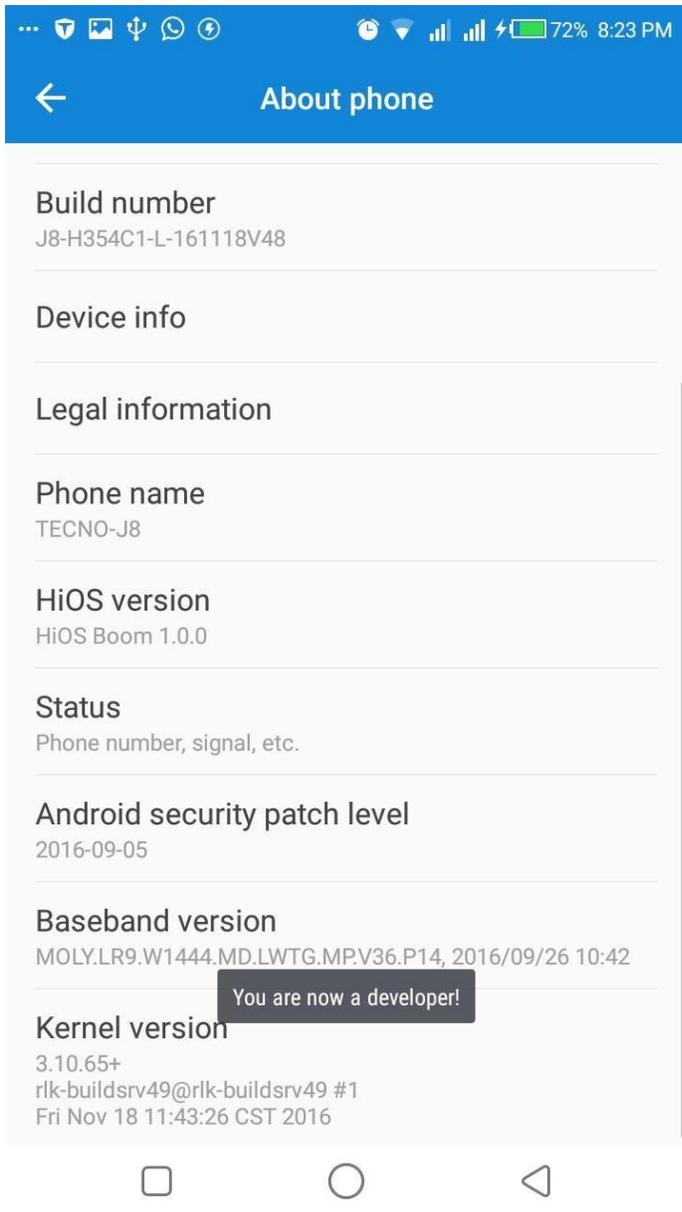


Figure 3.4 – About Phone section of the Android Settings app with “Developer Options” enabled

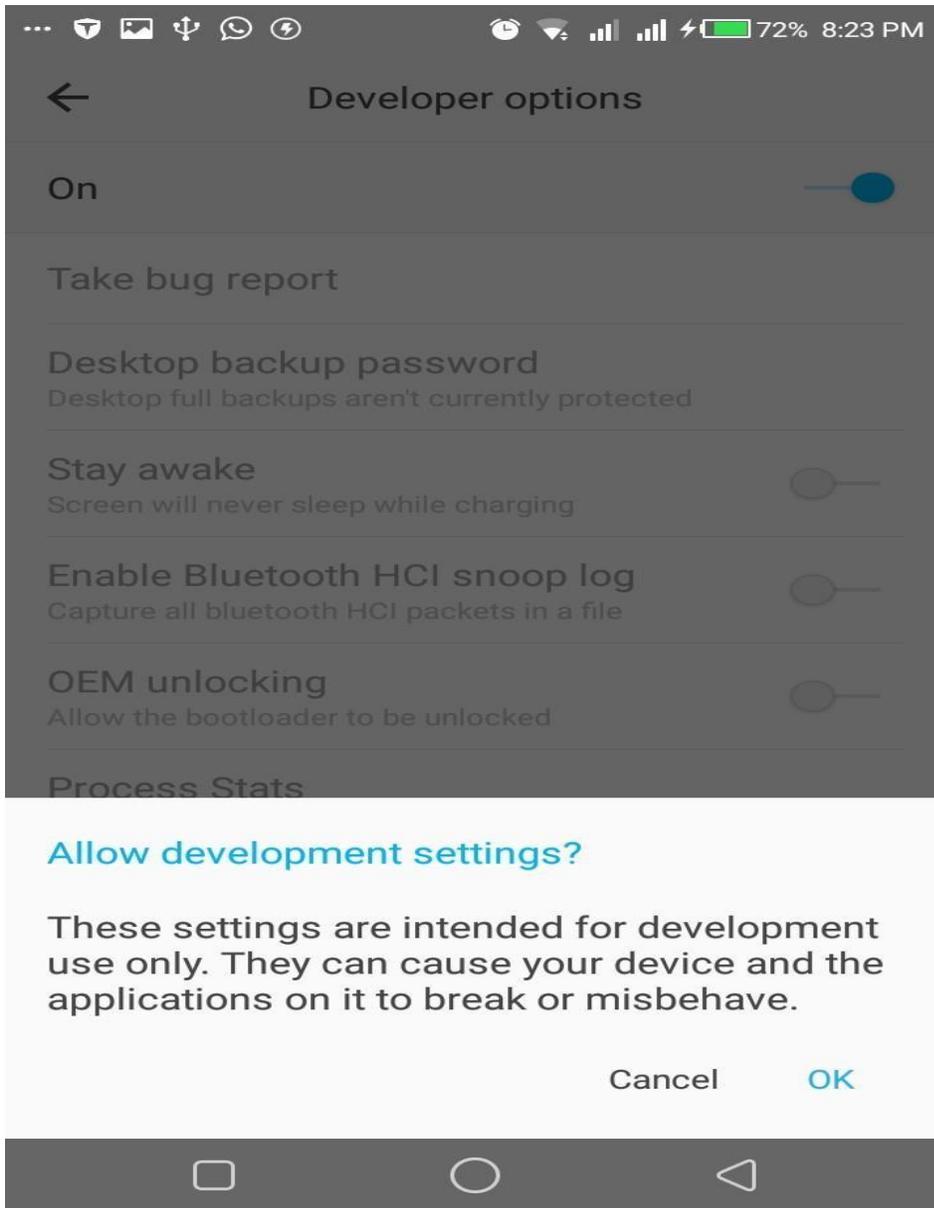


Figure 3.5 – Developer Options section of Android Settings

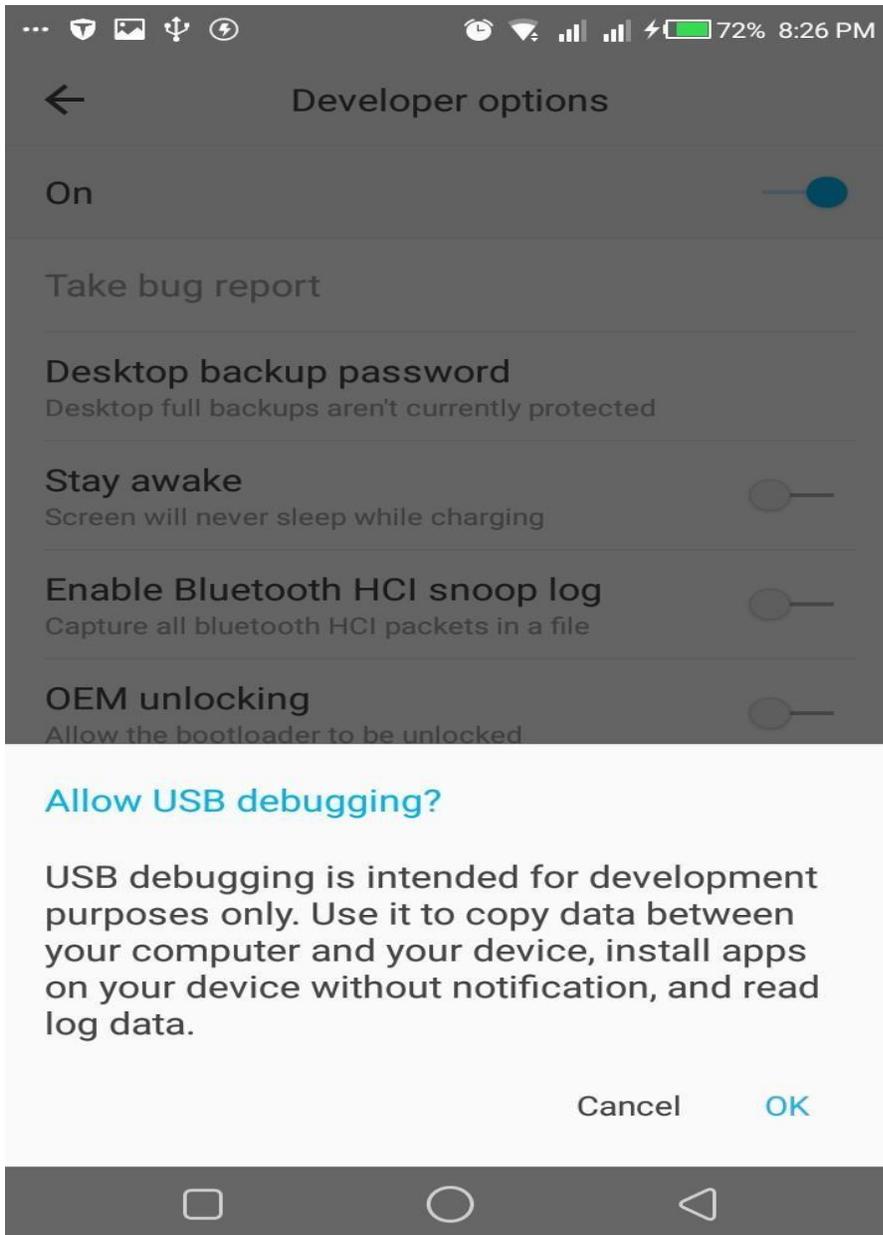


Figure 3.6 – Enabling USB Debugging on an Android device

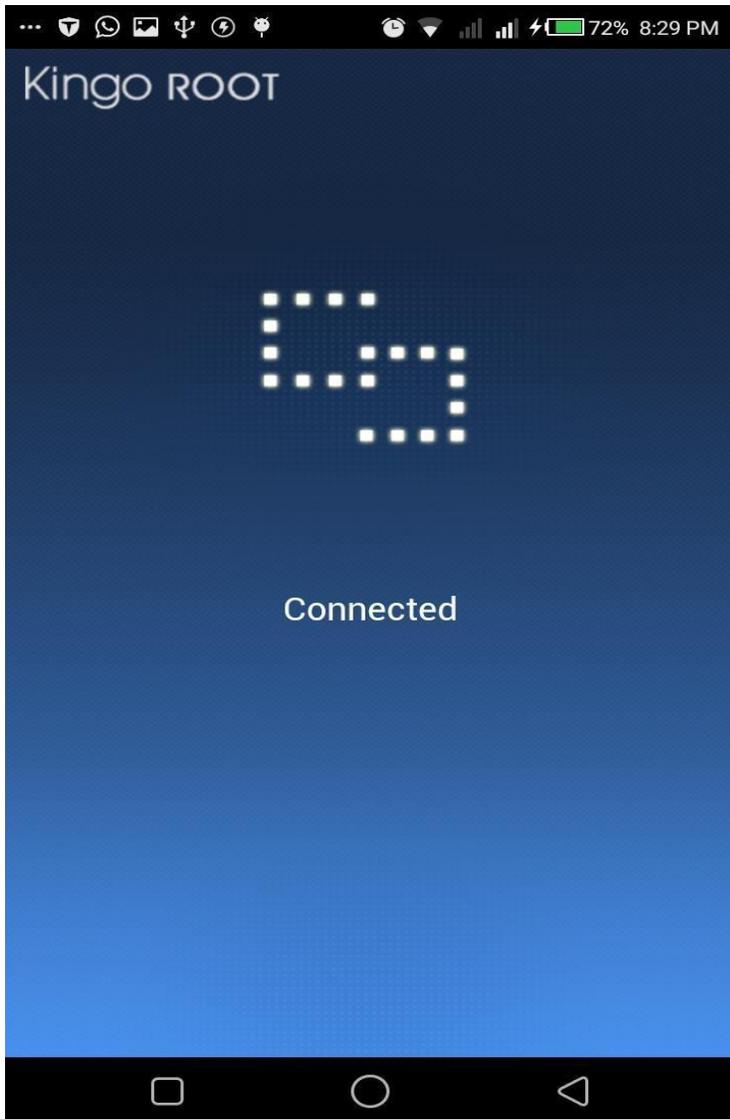


Figure 3.7 – Android screen showing connection to Kingo Root



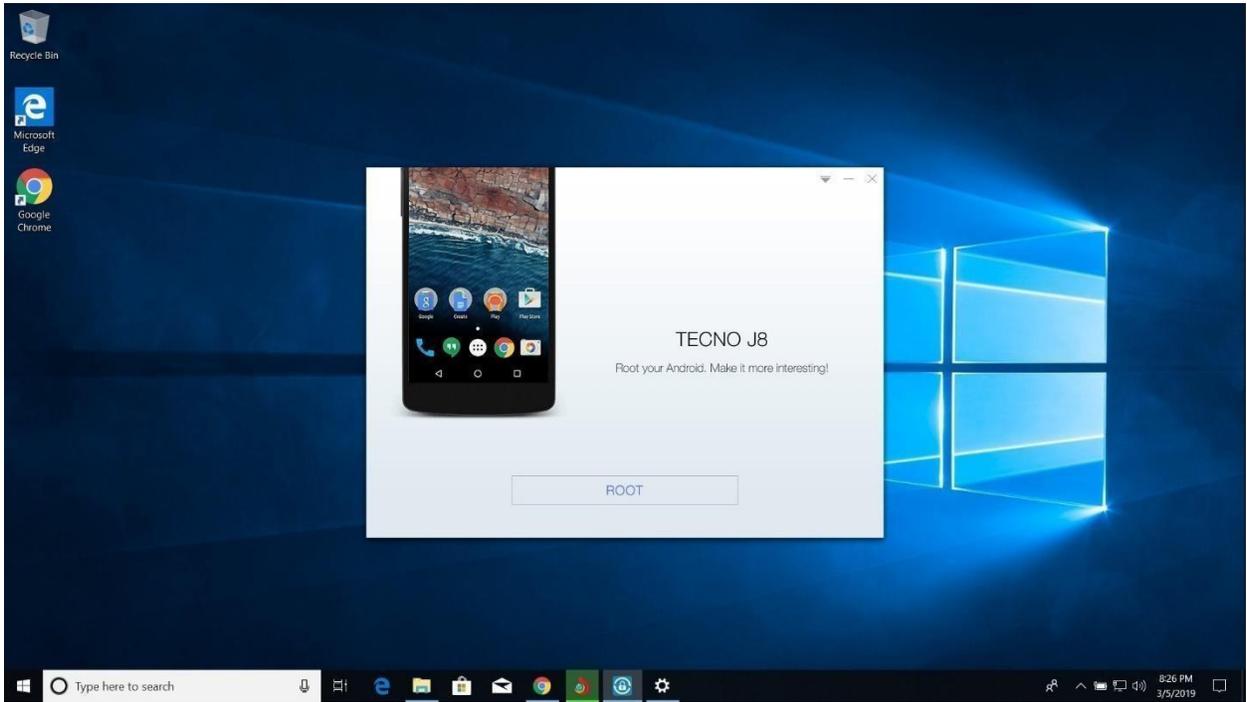


Figure 3.8 – Kingo Root application after a connection has been established with an unrooted Android device

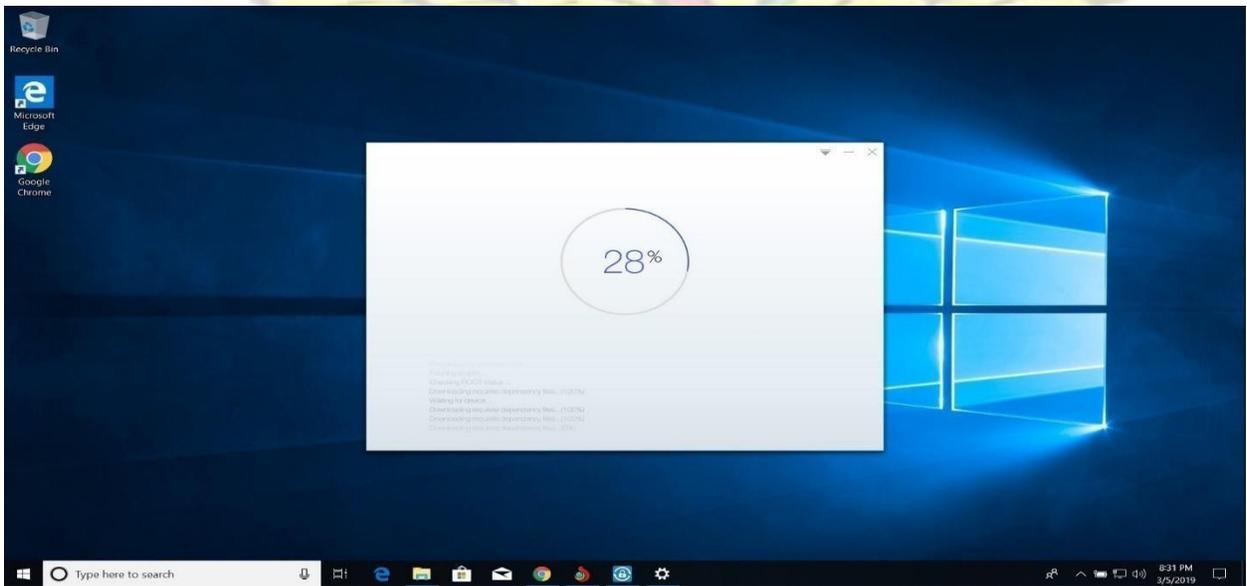


Figure 3.9 – Kingo Root desktop application showing rooting progress

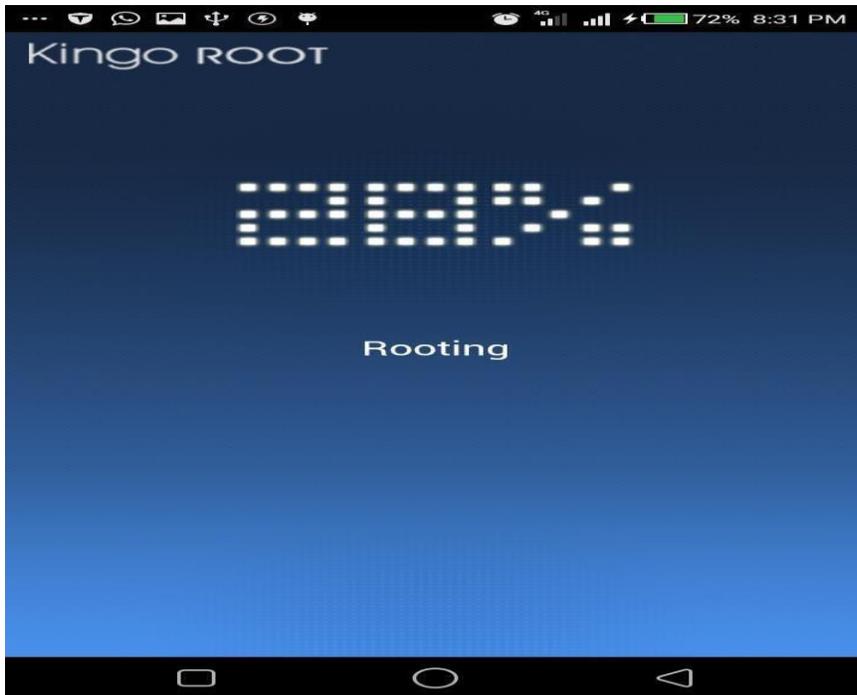


Figure 3.10 – Kingo Root progress on Android screen

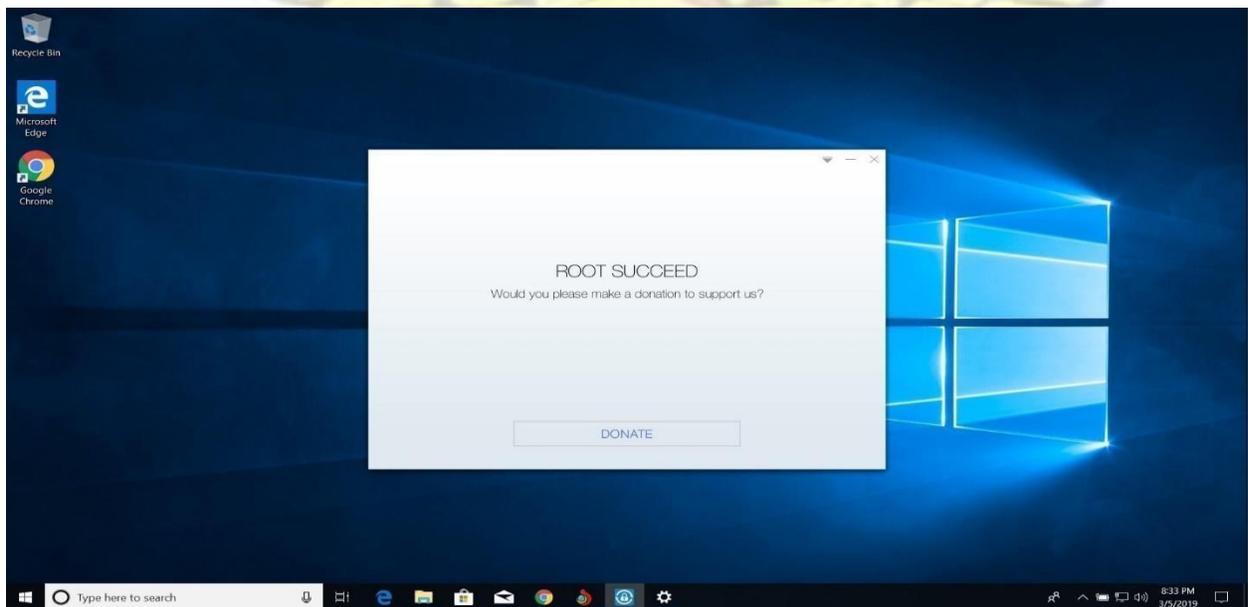


Figure 3.11 – Kingo Root application showing a successful rooting process

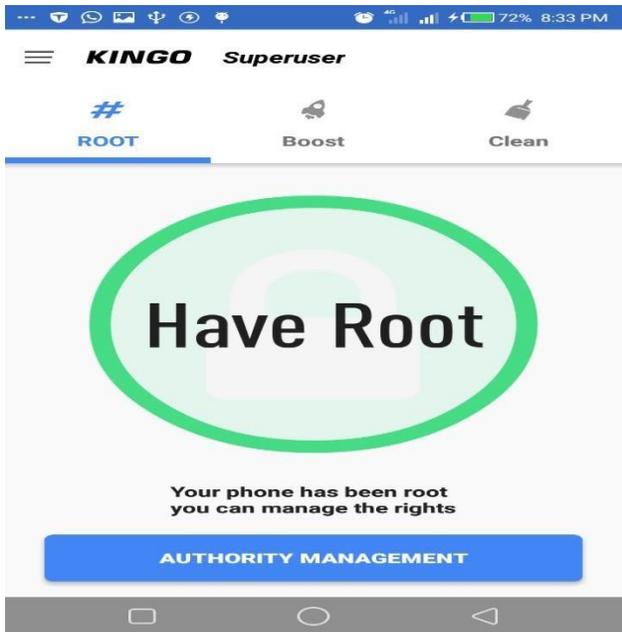


Figure 3.12 – Android phone screen after a successful rooting process.

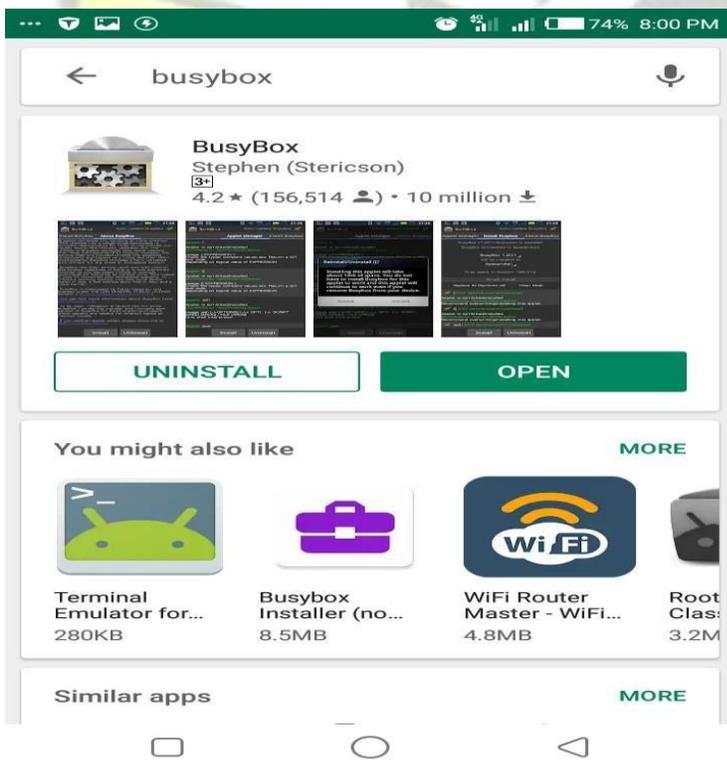


Figure 3.13 – Busybox application being installed from Google Play Store

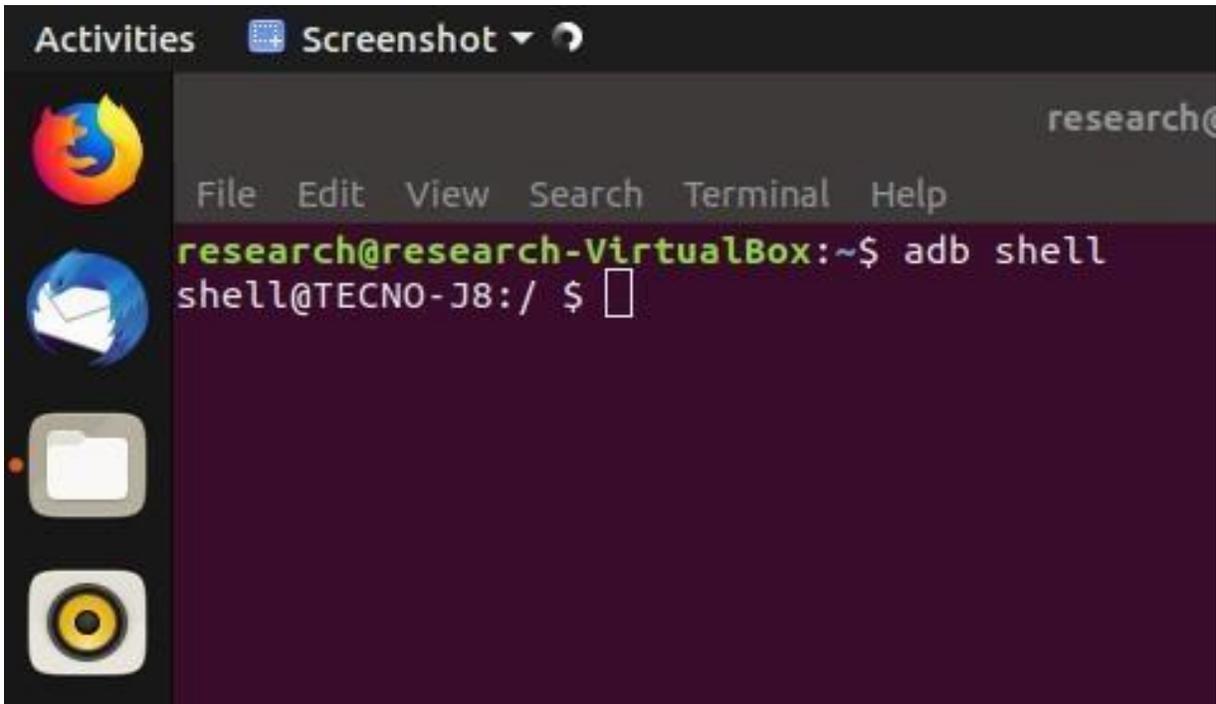


Figure 3.14 – ADB Shell Command and Output

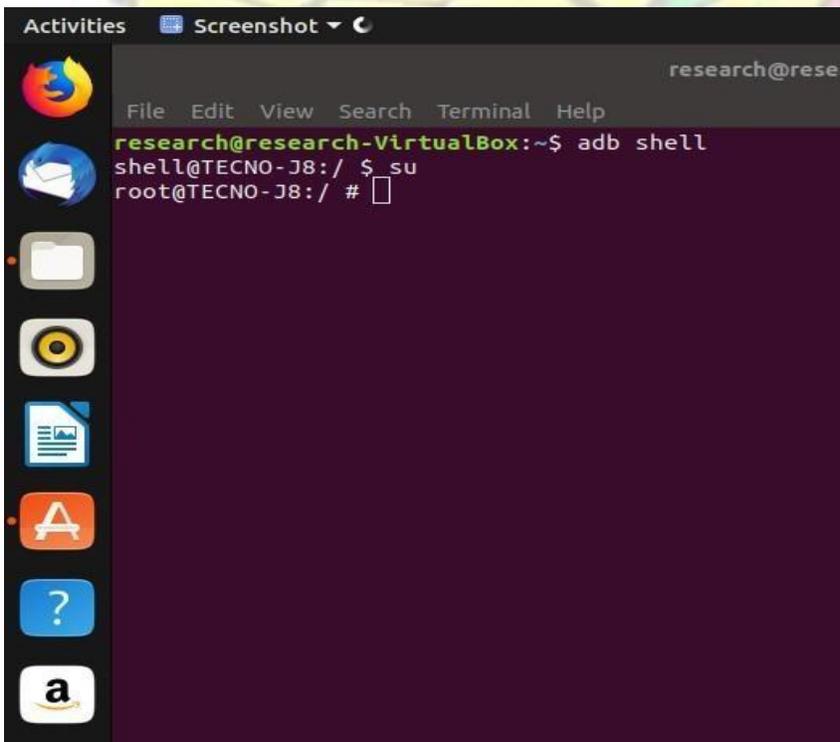


Figure 3.15 - SU command

```
Activities Screenshot research@research-V
File Edit View Search Terminal Help
research@research-VirtualBox:~$ adb shell
shell@TECNO-J8:/ $ su
root@TECNO-J8:/ # cat /proc/partitions
major minor #blocks name
254 0 524288 zram0
7 0 1254 loop0
179 0 15392768 mmcblk0
179 1 3072 mmcblk0p1
179 2 5120 mmcblk0p2
179 3 10240 mmcblk0p3
179 4 10240 mmcblk0p4
179 5 512 mmcblk0p5
179 6 512 mmcblk0p6
179 7 16384 mmcblk0p7
179 8 16384 mmcblk0p8
179 9 8192 mmcblk0p9
179 10 10240 mmcblk0p10
179 11 512 mmcblk0p11
179 12 2048 mmcblk0p12
179 13 6144 mmcblk0p13
179 14 8192 mmcblk0p14
179 15 5120 mmcblk0p15
179 16 5120 mmcblk0p16
179 17 1024 mmcblk0p17
179 18 32768 mmcblk0p18
179 19 37888 mmcblk0p19
179 20 3145728 mmcblk0p20
179 21 409600 mmcblk0p21
179 22 11640320 mmcblk0p22
179 23 16384 mmcblk0p23
179 96 4096 mmcblk0rpmb
179 64 4096 mmcblk0boot1
179 32 4096 mmcblk0boot0
179 128 7864320 mmcblk1
179 129 7863296 mmcblk1p1
root@TECNO-J8:/ #
```

Figure 3.16 – Output from "cat" command

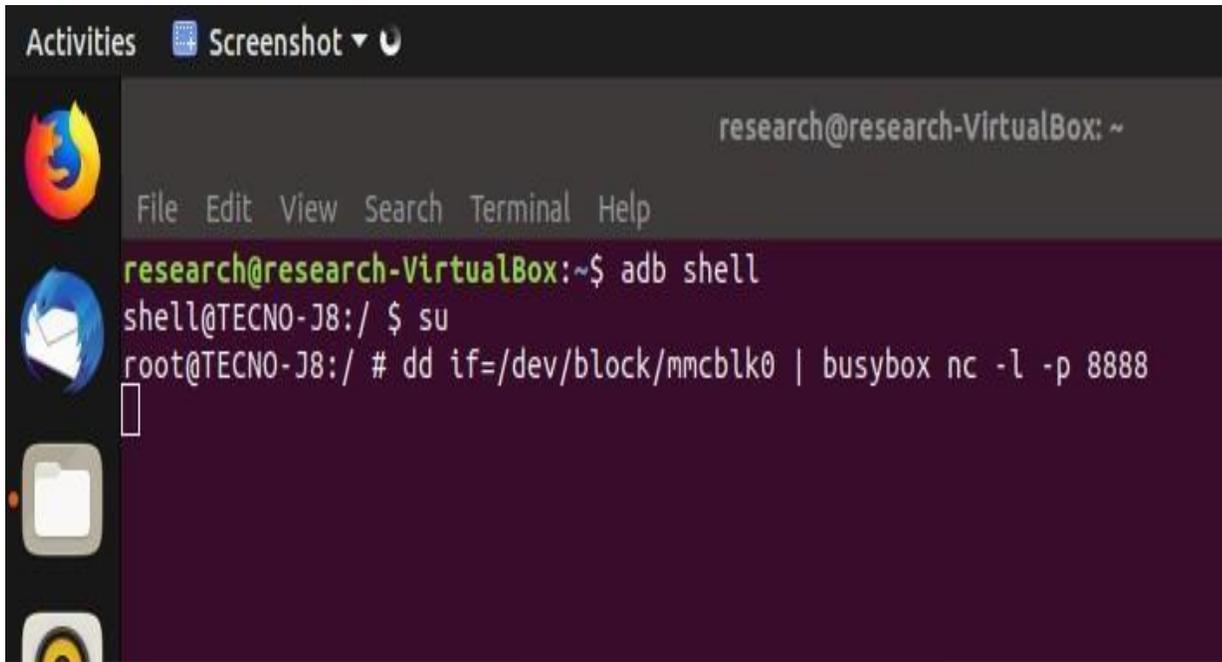


Figure 3.17 – DD commands

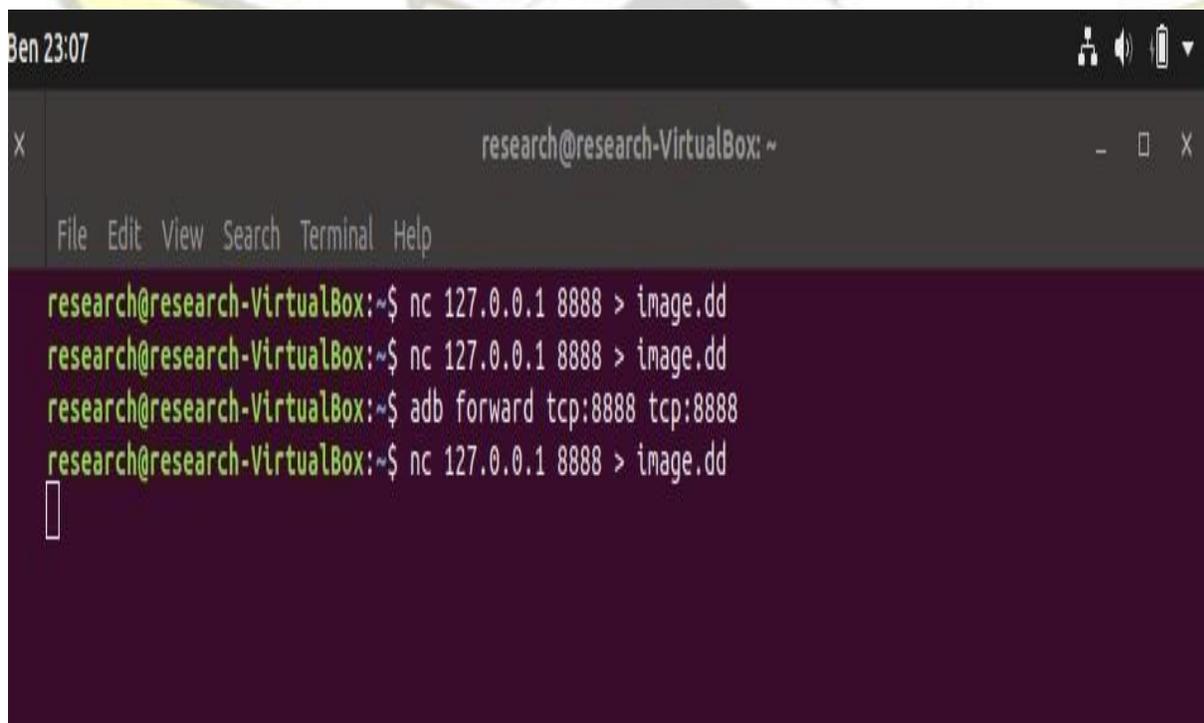


Figure 3.18 – NC command in Ubuntu

```
Activities Terminal Ben 22:03
research@research-Virtu
File Edit View Search Terminal Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
research@research-VirtualBox:~$ sudo apt-get install foremost
[sudo] password for research:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libncursesw5 libtinfo5
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  foremost
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 38.7 kB of archives.
After this operation, 99.3 kB of additional disk space will be used.
Get:1 http://gh.archive.ubuntu.com/ubuntu cosmic/universe amd64 foremost amd64 1.5.7-8 [38.7 kB]
Fetched 38.7 kB in 2s (18.1 kB/s)
Selecting previously unselected package foremost.
(Reading database ... 165590 files and directories currently installed.)
Preparing to unpack ../foremost_1.5.7-8_amd64.deb ...
Unpacking foremost (1.5.7-8) ...
Setting up foremost (1.5.7-8) ...
Processing triggers for man-db (2.8.4-2) ...
research@research-VirtualBox:~$
```

Figure 3.19 – Installation command and output for foremost

```
Activities Terminal Ben
research@research-VirtualBox:~$ sudo apt-get install scalpel
[sudo] password for research:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libncursesw5 libtinfo5
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  scalpel
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 28.8 kB of archives.
After this operation, 80.9 kB of additional disk space will be used.
Get:1 http://gh.archive.ubuntu.com/ubuntu cosmic/universe amd64 scalpel amd64 1.60-4 [28.8 kB]
Fetched 28.8 kB in 1s (30.6 kB/s)
Selecting previously unselected package scalpel.
(Reading database ... 165597 files and directories currently installed.)
Preparing to unpack .../scalpel_1.60-4_amd64.deb ...
Unpacking scalpel (1.60-4) ...
Setting up scalpel (1.60-4) ...
Processing triggers for man-db (2.8.4-2) ...
research@research-VirtualBox:~$
```

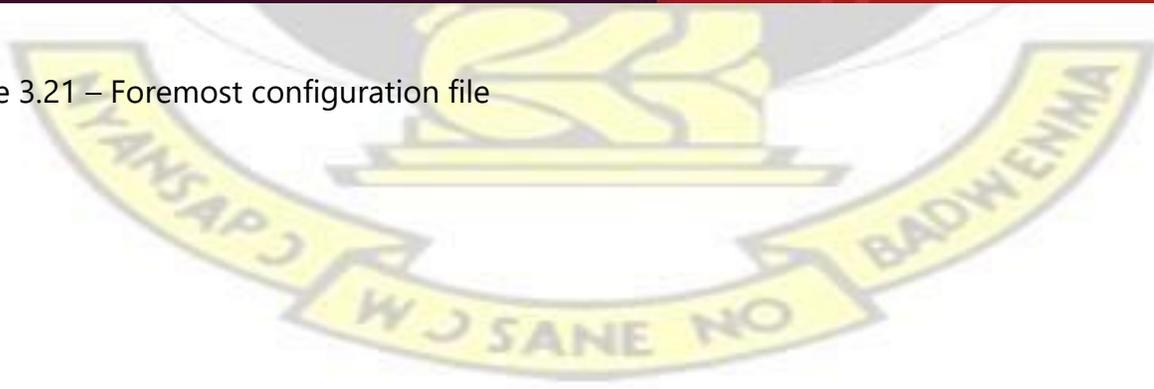
Figure 3.20 – Installation command and output for scalpel

```

Wuk 22:22
*foremost.conf
/etc
Open Save
#-----
# SOUND FILES
#-----
# (NOTE THIS FORMAT HAS A BUILTIN EXTRACTION FUNCTION)
# wav y 200000 RIFF????WAVE
#
# Real Audio Files
# ra y 1000000 \x2e\x72\x61\xfd
# ra y 1000000 .RMF
#
# asf y 8000000 \x30\x26\xb2\x75\x8E\x66\xCF\x11\xa6\xd9\x00\xaa\x00\x62\xCE\x6C
#
# wmv y 20000000 \x30\x26\xb2\x75\x8E\x66\xCF\x11\xa6\xd9\x00\xaa\x00\x62\xCE\x6C
#
# wma y 8000000 \x30\x26\xb2\x75 \x00\x00\x00\xFF
#
# wma y 8000000 \x30\x26\xb2\x75 \x52\x9A\x12\x46
#
# mp3 y 8000000 \xFF\xFB??\x44\x00\x00
# mp3 y 8000000 \x57\x41\x56\45 \x00\x00\xFF\
# mp3 y 8000000 \xFF\xFB\xD0\ \xD1\x35\x51\xCC\
# mp3 y 8000000 \x49\x44\x33\
# mp3 y 8000000 \x4C\x41\x4D\x45\
#-----
# WINDOWS REGISTRY FILES
#-----
#
# Windows NT registry
# dat y 4000000 regf
# Windows 95 registry
# dat y 4000000 CREG
#
# lnk y 5000 \x4C\x00\x00\x00\x01\x14\x02\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
# chm y 100000 \x49\x54\x53\x46\x03\x00\x00\x00\x60\x00\x00\x00\x01\x00\x00
# cookie n 4096 id=
# rdp y 4096
\xFF\xFE\x73\x00\x63\x00\x72\x00\x65\x00\x65\x00\x6E\x00\x20\x00\x6D
#
Plain Text Tab Width: 8 Ln 206, Col 1 INS

```

Figure 3.21 – Foremost configuration file



```
Activities Screenshot Wuk 22:28
research@research-VirtualBox: ~/Downloads
File Edit View Search Terminal Help
research@research-VirtualBox:~/Downloads$ hexdump Hillsong_Healer_HQ_HILLSONG.mp3 | head
00000000 4449 0433 0000 0000 0001 5854 5858 0000
00000010 1200 0000 6d03 6a61 726f 625f 6172 646e
00000020 6d00 3470 0032 5854 5858 0000 1100 0000
00000030 6d03 6e69 726f 765f 7265 6973 6e6f 3000
00000040 5400 5858 0058 0000 001c 0300 6f63 706d
00000050 7461 6269 656c 625f 6172 646e 0073 7369
00000060 6d6f 706d 3234 5400 5353 0045 0000 000f
00000070 0300 614c 6676 3635 332e 2e36 3031 0030
00000080 0000 0000 0000 0000 0000 fbff 64b0 0f00
00000090 00f0 6900 0000 0800 0000 200d 0000 0001
research@research-VirtualBox:~/Downloads$
```

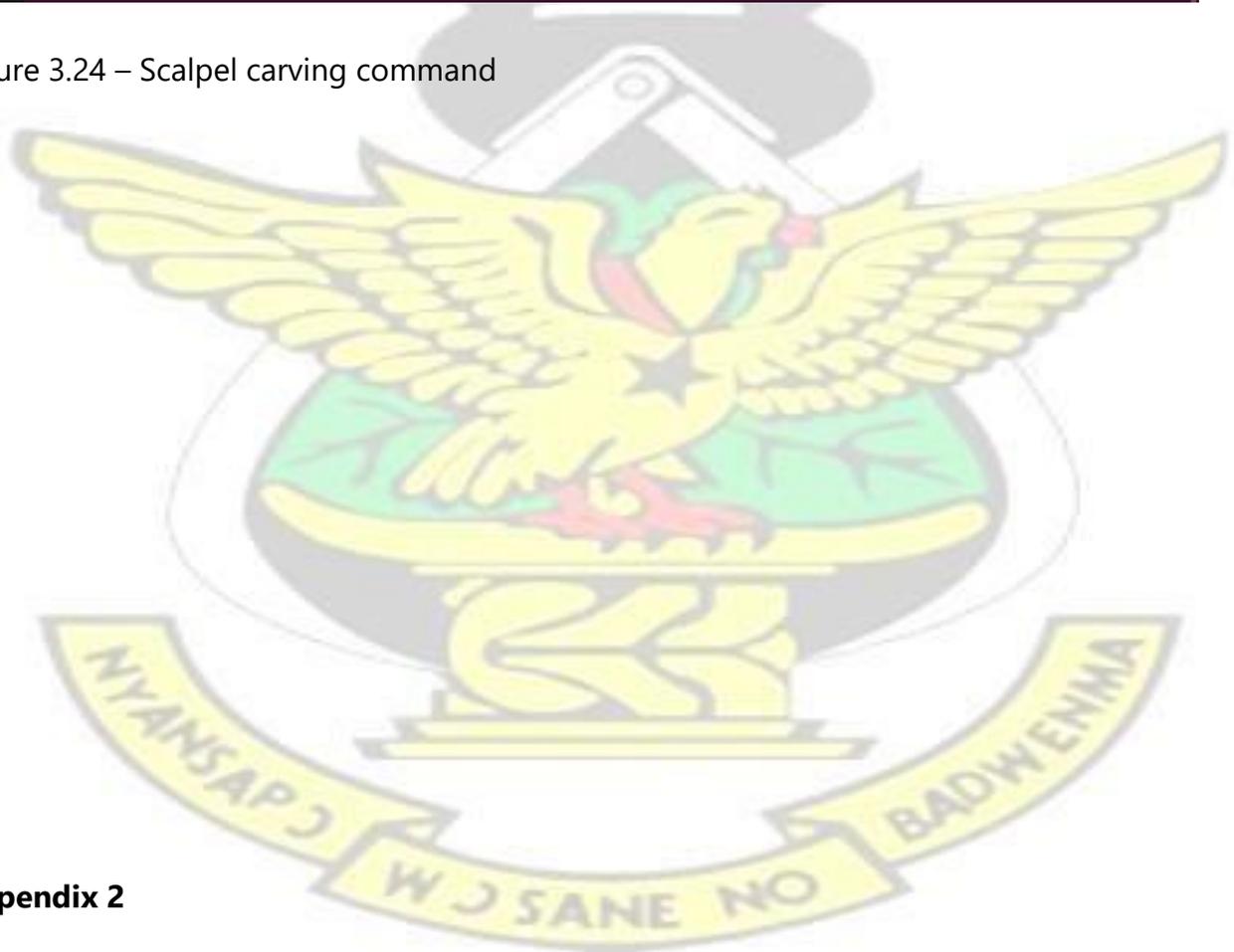
Figure 3.22 – Hexdump and Head commands

```
Activities Screenshot Dwo 20:46
research@research-HP-EliteDesk-800-G1-TWR: ~/Desktop/workspace
File Edit View Search Terminal Help
research@research-HP-EliteDesk-800-G1-TWR:~/Desktop/workspace$ mkdir foremost_output
research@research-HP-EliteDesk-800-G1-TWR:~/Desktop/workspace$ foremost -o foremost_output/ image.dd
Processing: image.dd
|*****|
research@research-HP-EliteDesk-800-G1-TWR:~/Desktop/workspace$
```

Figure 3.23 – Foremost carving command.

```
Activities Screenshot ▾ Dwo 21:01
research@research-HP-EliteDesk-800-G1-TWR: ~/Desktop/workspace
File Edit View Search Terminal Help
research@research-HP-EliteDesk-800-G1-TWR:~/Desktop/workspace$ mkdir scalpel_output_2
research@research-HP-EliteDesk-800-G1-TWR:~/Desktop/workspace$ scalpel -o scalpel_output_2/ image.dd
Scalpel version 1.60
Written by Golden G. Richard III, based on Foremost 0.69.
Opening target "/home/research/Desktop/workspace/image.dd"
Image file pass 1/2.
image.dd: 100.0% |*****| 5.2 GB 00:00 ETA
Allocating work queues...
Work queues allocation complete. Building carve lists...
Carve lists built. Workload:
jpg with header "\xff\xd8\xff\xe0\x00\x10" and footer "\xff\xd9" --> 384 files
db with header "\x53\x51\x4c\x69\x74\x65\x20\x66\x6f\x72\x6d\x61\x74\x20\x33" and footer "" --> 271 files
Carving files from image.
Image file pass 2/2.
image.dd: 100.0% |*****| 5.2 GB 00:00 ETA
Processing of image file complete. Cleaning up...
Done.
Scalpel is done, files carved = 655, elapsed = 48 seconds.
research@research-HP-EliteDesk-800-G1-TWR:~/Desktop/workspace$
```

Figure 3.24 – Scalpel carving command



## Appendix 2

Thank you for your acceptance to participate in this study. This is a survey on **Knowledge about Android**

**Device Security.** Notice that the information you share will only be used for academic purposes and will be treated as strictly confidential as practicable. For any clarification or queries about this questionnaire, kindly contact +233244263434 or [kpeasah@gmail.com](mailto:kpeasah@gmail.com).

**PART 1: DEMOGRAPHIC AND SOCIOECONOMIC CHARACTERISTICS**

Please indicate your response by ticking [√] the appropriate box

S/N	Questions & Filters	Responses	Code
1	Which age group are you?	<input type="checkbox"/> ≤ 17 years <input type="checkbox"/> 18-24 years <input type="checkbox"/> 24-30 years <input type="checkbox"/> ≥31 years	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4
2	What is your sex?	<input type="checkbox"/> Male <input type="checkbox"/> Female	<input type="checkbox"/> 0 <input type="checkbox"/> 1
3	What is your educational level?	<input type="checkbox"/> No education <input type="checkbox"/> Basic education <input type="checkbox"/> SHS/Technical/Vocational <input type="checkbox"/> Tertiary	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4
4	What type of phone do you use currently?	<input type="checkbox"/> Android phone <input type="checkbox"/> Windows phone <input type="checkbox"/> Apple <input type="checkbox"/> Non-smart phone (yam)	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4

**PART 2: PHONE USAGE AND ANDRIOD SECURITY**

S/N	Questions & Filters	Responses	Code
5	Which of these information do you usually access/store on your phone? (Select all applicable)	<input type="checkbox"/> Email <input type="checkbox"/> Financial Transactions <input type="checkbox"/> Health information <input type="checkbox"/> Academic information <input type="checkbox"/> Social information (work, family, friends, etc.) <input type="checkbox"/> Social media (facebook, twitter, etc) <input type="checkbox"/> Private pictures/Videos <b>Tick all that apply</b>	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7
6	Have you ever owned an Android phone before your current phone	<input type="checkbox"/> Yes <input type="checkbox"/> No	<input type="checkbox"/> 0 <input type="checkbox"/> 1
7	If yes to <b>Q 6</b> , why did you stop using it/them?	<input type="checkbox"/> It got spoiled <input type="checkbox"/> I wanted a better phone <input type="checkbox"/> It was outmoded <input type="checkbox"/> It got lost <input type="checkbox"/> Others .....	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4

Correspondence: Kwame Ofosuhene Peasah, Lecturer, Department of Computer Science, KNUST.



8	Which phone model were you using?	Samsung Huawei LG	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3
		Infinix Tecno Others .....	<input type="checkbox"/> 4 <input type="checkbox"/> 5
9	How did you dispose of the phone?	Sold Gift Exchange Still Keeping Missing/Stolen	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5
10	What data or information on your phone do you consider to be private, or you don't want other people (especially strangers) to see?	Photos Videos Audios Office Documents (Word Excel, PPT, PDF) Chat history/Notes Contacts Call history Browser history, bookmarks and Cache <b><i>Tick all that apply</i></b>	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8
11	Do you have a google account (gmail account)?	Yes No	<input type="checkbox"/> 0 <input type="checkbox"/> 1
12	If yes, do you log into your account on your phone (synchronise)?	Yes No	<input type="checkbox"/> 0 <input type="checkbox"/> 1

S/ N	Questions & Filters	Responses				
		Least likely	Likely	Do Nothing	Certain	Very certain
	How do you get rid of those private data before disposing of your phone?	1	2	3	4	5
13	Delete specific items from the device	<input type="checkbox"/>				
14	Format the memory card (sdcard)	<input type="checkbox"/>				
15	Perform a factory reset on the phone	<input type="checkbox"/>				

16	Use a "secure deletion" app to erase the phone memory/memory card	[ ]	[ ]	[ ]	[ ]	[ ]
----	---	-----	-----	-----	-----	-----

# KNUST



17	Encrypt the phone, then reset	[ ]	[ ]	[ ]	[ ]	[ ]
----	-------------------------------	-----	-----	-----	-----	-----

S/N	Questions & Filters	Responses	Code
18	Do you know of encryption of Android phones?	Yes No	[ ] 0 [ ] 1
19	If yes in <b>Q 18</b> , have you ever used it?	Yes No	[ ] 0 [ ] 1
20	Was there a password or pattern on your phone while in use?	Yes No	[ ] 0 [ ] 1
21	Are you aware that data from an Android phone can be accessed after it has been deleted or the device has been reset?  <b>If no, stop</b>	Yes No	[ ] 0 [ ] 1
22	Have you tried accessing your deleted files before?	Yes No	[ ] 0 [ ] 1
23	What types of files did you access?	Pictures Office documents Notes/chat history Contacts	[ ] 1 [ ] 2 [ ] 3 [ ] 4
24	By what means did you retrieve your deleted files?	Used Android apps Used cloud back-up Others	[ ] 1 [ ] 2 [ ] 3

# KNUST

## APPENDIX 3 – HEADLESS BLOCK SWAPPING SIMULATION CODE

File System Snapshot Module

```
#bin/bash

#read previous_snapshot=()
current_snapshot=()
navi=records

getPath(){
    #hold the path to the folder which contains inode files and pass it to next function
    listFiles
}

listFiles(){

    #recieve folder path, list all files in folder and find last file    allFiles=`ls
$navi`    #get length of files in folder    arr_length=$(( ${#allFiles[*]} )
#get the last file(Current)    currentFile_index=$(( arr_length-1 ))    #get the
```

```
last but one (previous)  previousFile_index=${arr_length-2}  #funciton  
to read contents of file  readFile  
"$navi/${allFiles[$previousFile_index]}"
```

# KNUST

```
$currentFile_index
```

```
readFile(){
```

```
    #reads last index of previous file line by line else it belong to current snapshot
```

```
while read -r line; do    index=${arr_length-2}
```

```
    if [ $2 -eq $index ]
```

```
        previous_snapshot="{previous_snapshot[@]}" "$line"
```

```
    else
```

```
        current_snapshot="{current_snapshot[@]}" "$line"
```

```
done < "$1"
```

```
compareSnapshots(){
```

```
    #initialize array for deleted paths deletedPaths=()
```

```
    #compare each file in previous snapshot to current snapshot
```

```
for prev in "${previous_snapshot[@]}" do
```

```
    del=()
```

```
    found=0    for current in
```

```
        "${current_snapshot[@]}"
```

```
$previousFile_index    readFile "$navi/${allFiles[$currentFile_index]}"
```

```
}
```

```
then
```

```
fi
```

```
}
```

```
do
```

```
#if file is found set found variable to 0 else not found so add to deleted paths
```

```
if [ "$prev" = "$current" ]
```

```
then
```

# KNUST

```
        found=1
break    else
        #echo $prev
del=$prev
    fi
done    if [ $found -eq 0 ] then
deletedPaths=("${deletedPaths[@]}" "$del")
```

```
#loop through deleted path and passing whole file information to forprint function
```

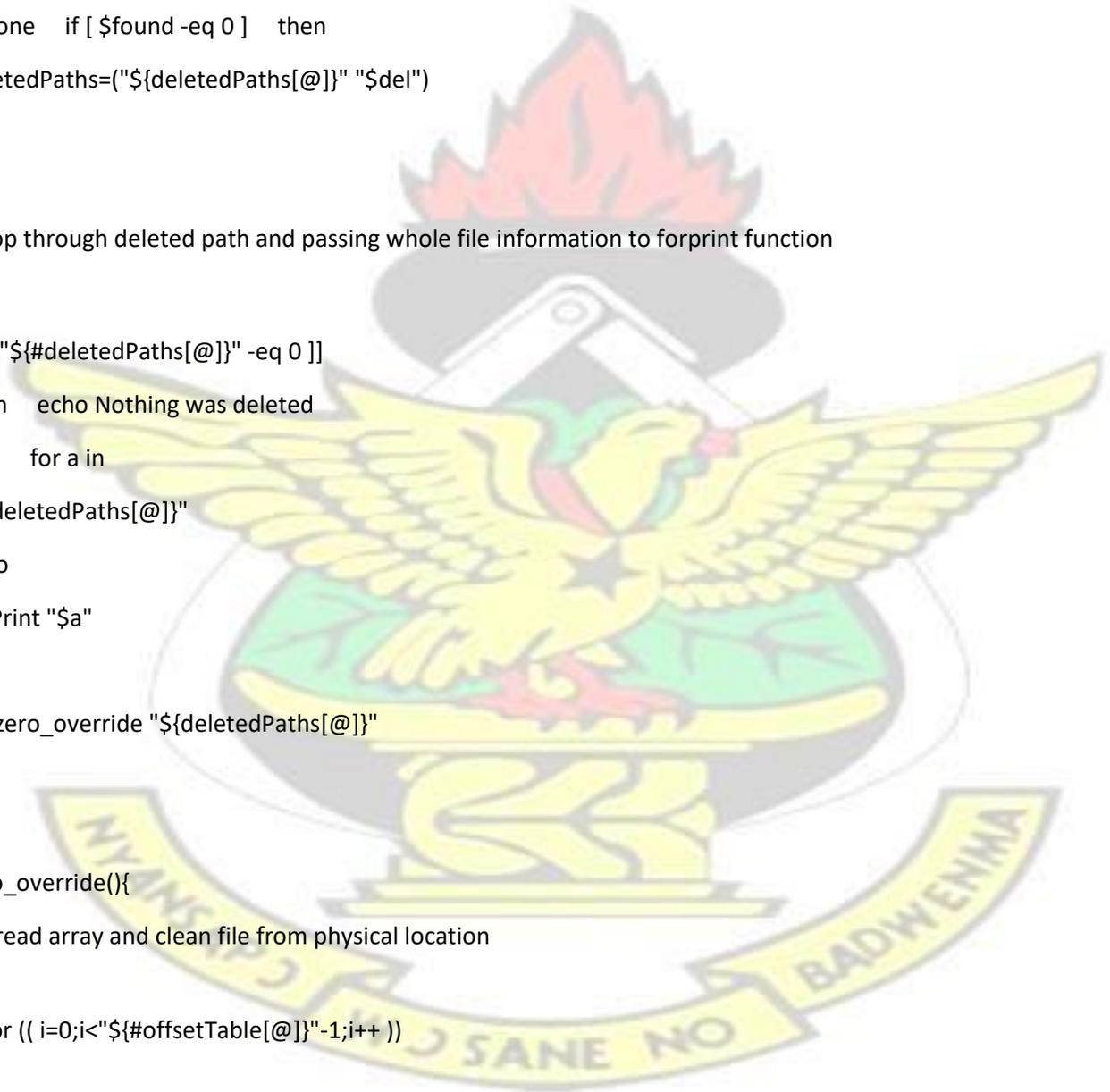
```
if [ "${#deletedPaths[@]}" -eq 0 ]
then    echo Nothing was deleted
else    for a in
"${deletedPaths[@]}"
do
forPrint "$a"

#zero_override "${deletedPaths[@]}"
```

```
zero_override(){
#read array and clean file from physical location
```

```
for (( i=0;i<"${#offsetTable[@]}"-1;i++ ))
```

```
fi done
```



done

fi

}

do

echo

```

echo The Memory location contains:

phys_add="{offsetTable[$i,0]}"
size=$(( "{offsetTable[$i,1]}" ))
count=$(( $size * 4096 ))
input="/dev/zero"    output="/dev/sda"
ibs=1
skip=$(( 2048 * 512 + $phys_add * 4096 ))

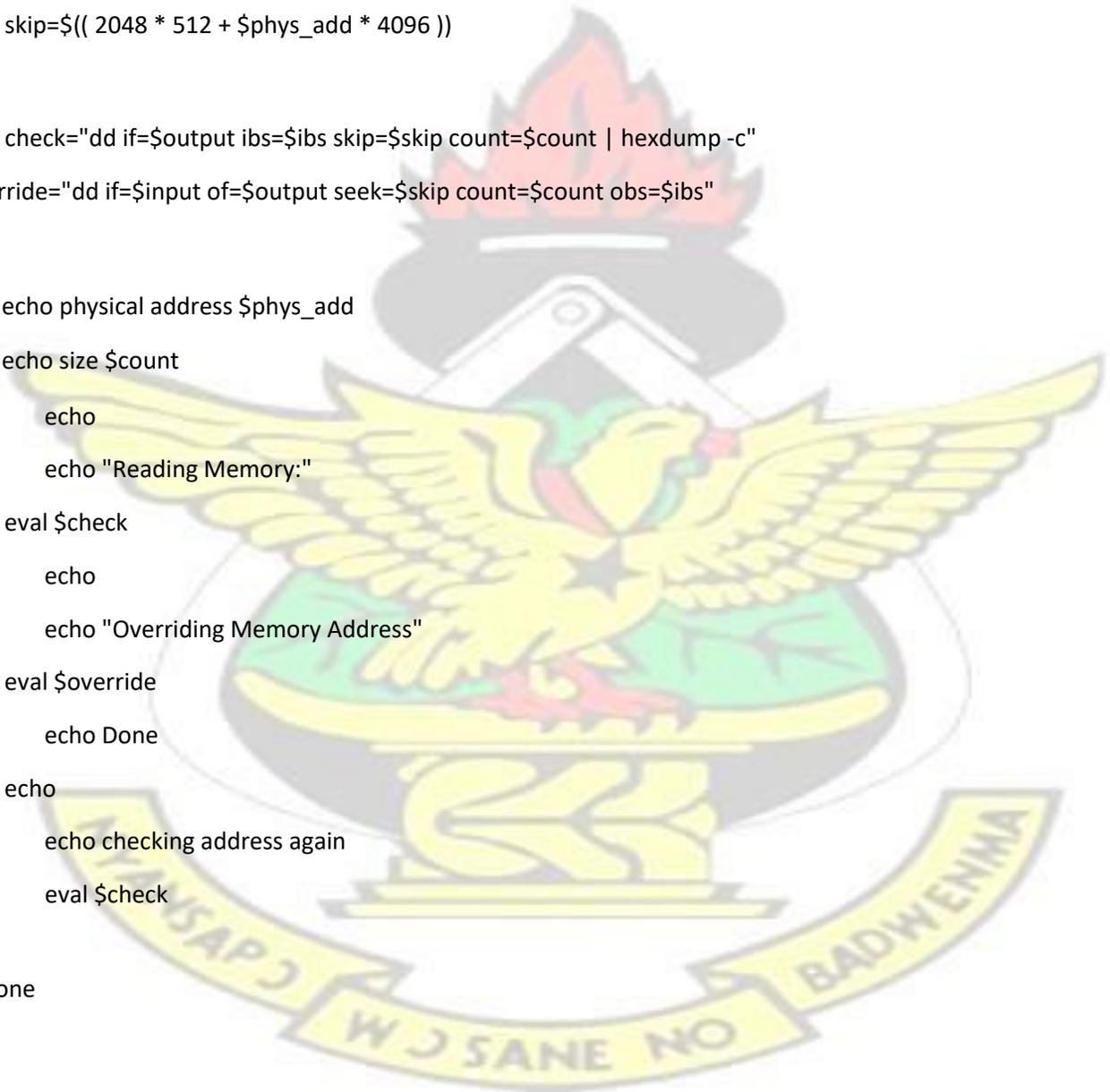
check="dd if=$output ibs=$ibs skip=$skip count=$count | hexdump -c"
override="dd if=$input of=$output seek=$skip count=$count obs=$ibs"

echo physical address $phys_add
echo size $count
echo
echo "Reading Memory:"
eval $check
echo
echo "Overriding Memory Address"
eval $override
echo Done
echo
echo checking address again
eval $check

done

```

# KNUST



```
}
```

```
-A offsetTable
```

```
awkseive='{ $1="" ; $2="" ; $3="" ; $4="" ; $5="" ; $6="" ; print }'
```

```
arr("$@")  physical_offset=()  counterA=0
```

```
counterB=0  deletedSize=$(echo $a | awk '{print $4}')
```

```
#storing file data in multidimensional array  for i in
```

```
`echo "${arr[@]}" | awk $awkseive`
```

```
do  if [[ $i = "[" || $i = "]" || $i =  
", " ]]  then
```

```
:
```

```
elif [[ $i = ", " ]]
```

```
then
```

```
((counterA++))
```

```
counterB=0
```

```
offsetTable[$counterA,$counterB]=$i
```

```
((counterB++))
```

```
fi
```

```
done
```

```
#just echoing file information ( path and size)
```

```
echo  echo File information:
```

```
echo "${arr[@]}" | awk '{print "\tPath: " $2}'
```

```
echo $deletedSize | awk '{print "\tSize: "$0}'
```

```
#so if size of array below is 16 it means 8 offsets and 8 lengths and can be referenced
```

```
#to print data forPrint(){  #initialising variables  declare else
```



```
#example of usage "${offsetTable[1,1]}" echo Size of 2D
```

# KNUST

```
getPath compareSnapshots
```

```
Deletion Detection
```

```
#bin/bash
```

```
get_data(){
```

```
    phys_add=1099900
```

```
    skip=$(( 2048 * 512 + $phys_add * 4096 ))
```

```
    output="/dev/sda"
```

```
    count=4096
```

```
    dummy="$(dd if=$output ibs=1 skip=$skip count=$count)"
```

```
check_data(){
```

```
    echo
```

```
    echo scanning memory data one
```

```
    data_one=get_data
```

```
    echo
```

```
    echo waiting before next scan
```

```
array: ${#offsetTable[@]} | awk '{print "\t"$0}' zero_override
```

```
echo
```

}

}

```
sleep 10 }  
echo
```

```
echo scanning memory data two
```

```
data_two=get_data
```

```
echo
```

```
if [ $data_one = $data_two ]
```

```
then
```

```
    echo Data persists. Delete.
```

```
    #the line below will delete the data from memory address if uncommented #override=$(dd if=$input  
of=$output seek=$skip count=$count obs=$ibs)"
```

```
    echo
```

```
fi
```

```
check_data
```

### **Data Sanitization (Decapitation and Block Swapping) Module**

```
#bin/bash
```

```
filePaths=()
```

```
thisTime=() table=()
```

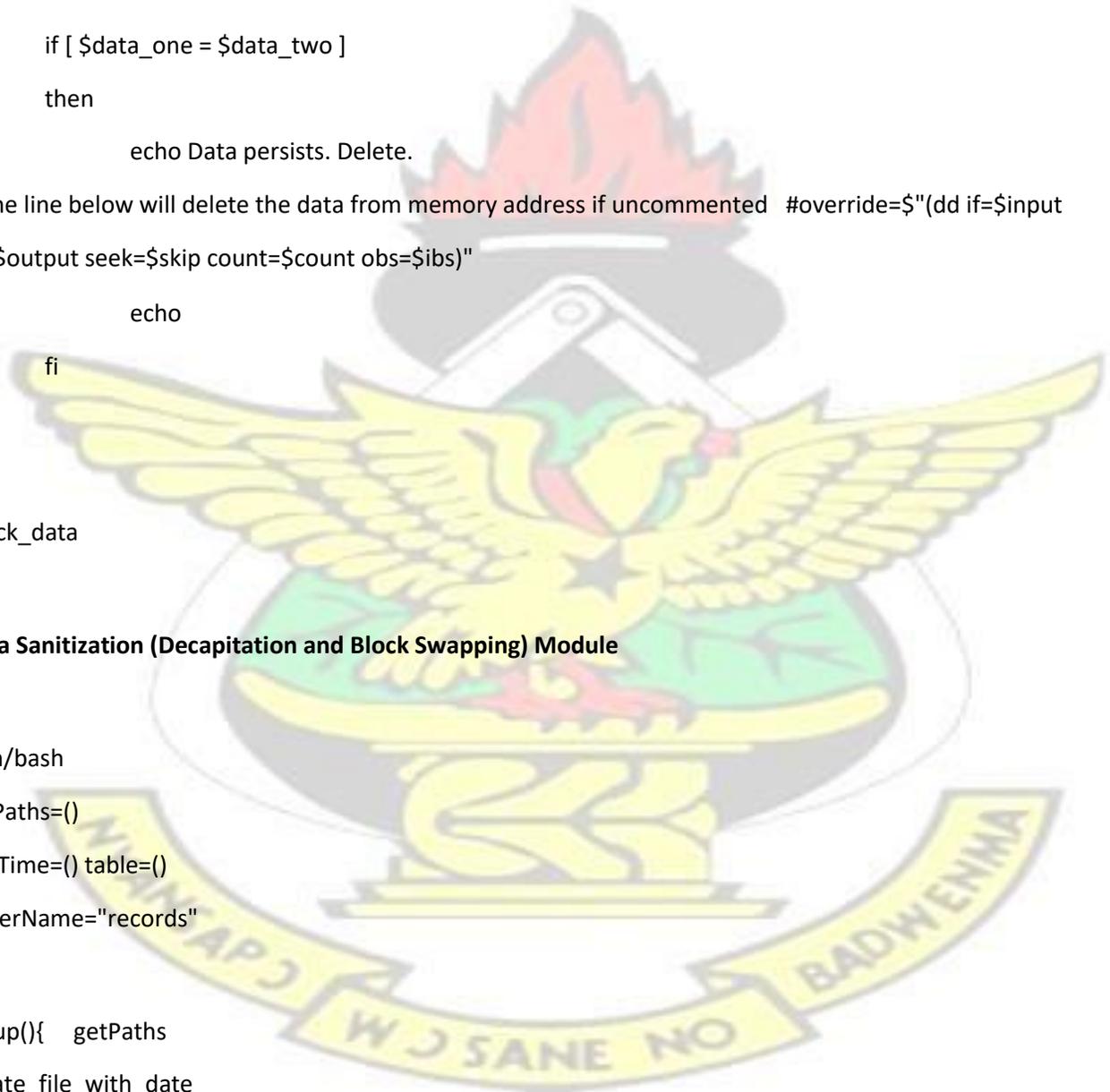
```
folderName="records"
```

```
group(){ getPaths
```

```
create_file_with_date
```

```
writing $thisTime
```

# KNUST



}

```

#Listing all files in current directory(including sub directories) getPaths(){
    filePaths=(`find *`)
arr_length=$(( ${#filePaths[*]} ))
    # echo "${#filePaths[@]}"

getData(){
    inode=$(stat $1 | grep Inode: | awk '{print $4}')    size=$(stat $1 | grep Size: | awk '{print
$2}')    physical_offset=$(filefrag -v $1 | awk 'FNR==4' | awk '{print substr($5, 0, (length($5)))}')
    physical_off=(`filefrag -v $1 | awk 'FNR>3' | awk '{print hi substr($4, 0, (length($4)-1) )}' | head -n -1`)
    offset_length=(`filefrag -v $1 | awk 'FNR>3' | awk '{print substr($6,0,(length($6)))}' | head -n -1`)

    # echo this is "${offset_length[@]}"
if [ "$physical_offset" = "" ]    then
physical_offset="null"

    if [ "$size" = "0" ]
then
size="null"

    #offSet_length_separator "${physical_off[@]}"
offSet_length_separator $1

}

```

fi

fi

}

#Write to file writing(){ #check if folder "records" exists if not create check\_folder\_exists for each in

```

else
"${filePaths[@]}"
do  getData `pwd`/"$each" # echo "${#table[@]}" if [
"${#table[@]}" -eq 0 ] then  echo [ `pwd`/"$each" , $size ] , [
null , null ]>> $folderName/$1

    echo [ `pwd`/"$each" , $size ] , "${table[@]}">> $folderName/$1
}

table=()
done
}

#creates file with date as name
create_file_with_date(){
thisTime=(`date +%s`)
then
:
else
fi }

check_folder_exists(){
if [ -d $folderName ]

mkdir $folderName

offSet_length_separator(){  for (( i=0;i<="${#physical_off[@]}"-1;i++))

```

# KNUST



```
do    echo [ "${physical_off[$i]}" , "${offset_length[$i]}" ]
table=("${table[@]}" [ "${physical_off[$i]}" , "${offset_length[$i]}" ],)
```

```
done
}
echo
```

```
echo Taking First Snapshot echo "[
Address, Length ]" group echo echo
waiting time before next snapshot...
sleep 10 echo echo Taking
Second Snapshot echo "[
Address, Length ]" group
```



# KNUST

## Appendix 4

Published articles from the study and reviewers' comments.

1. Kwame Peasah, Michael Asante, Osei Adjei, Darko Williams (2020). Accessing the Degree of Vulnerability of Android Devices Arising from Deleting Flaws Using Open Source Forensic Tools and Addressing the Vulnerability Using Headless Block Swapping (HBS). Information Technology for development
2. Kwame Ofosuhene Peasah, Michael Asante (2018). Awareness and knowledge about Android smartphones security among Ghanaians. International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 05
3. K. O. Peasah, Ebenezer Quayson, Osei Agyei, Ed. Danso Ansong 2017). Survey of Digital Forensic Models and Proposed Thematic Scheme. International Journal of Computer Applications (0975 – 8887)

# KNUST

