COPYRIGHT AND CITATION CONSIDERATIONS FOR THIS THESIS/ DISSERTATION

- Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

- NonCommercial — You may not use the material for commercial purposes.

- ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

How to cite this thesis
INTRUSION DETECTION FOR UNSOLICITED SHORT-MESSAGE SERVICES IN MOBILE DEVICES

OLORUNFEMI TOPE ROSELINE

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING SCIENCE

FACULTY OF ENGINEERING AND BUILT ENVIRONMENT

UNIVERSITY OF JOHANNESBURG

SOUTH AFRICA

A dissertation submitted in partial fulfilment of the requirements for the Master’s degree in Engineering in Electrical and Electronic Engineering

Department of Electrical and Electronic Engineering Science

University of Johannesburg, South Africa

Supervisor

Dr Nnamdi Nwulu

Co-Supervisor

Prof. Bheki sipho Twala

October, 2017
DEDICATION

This work is dedicated to the Almighty God.
COPYRIGHT STATEMENT

The copy of this dissertation has been supplied; on condition that anyone who consults it is understood to recognize that its copyright rests with the University of Johannesburg, South Africa; and that no information derived from it may be published without the author’s prior consent, unless correctly referenced.
DECLARATION

I, Tope Roseline Olorunfemi, hereby declare that the research work documented in this dissertation is my own; and no portion of the work has been submitted in support of an application for the awarding of another degree or qualification at this, or any other university, or other institute of higher learning. All sources used or cited have been correctly documented and fully acknowledged.

----------------------------------------
Date:

----------------------------------------
Author’s Signature
ACKNOWLEDGMENTS

I would like to first acknowledge my Creator for granting me the grace, wisdom and enablement to carry out this research work. I am indeed grateful Lord Jesus!

My appreciation goes to my supervisor, Dr Nnamdi Nwulu, for his unrelenting and detailed supervision during this research work. I appreciate you for your input and the moral support: you are the best supervisor!!!

My profound gratitude also goes to my parents, Chief Engr Paul Dare Olorunfemi (late) and Prophetess Felicia Shola Olorunfemi, for their unending love, guardianship and academic input given to me especially Dad. I am really proud to be one of your kids. How I wish Dad was still here with us; I miss you Dad. To my wonderful children, Prince and Opeyemi, for coping well - even when I am far away from home. I love you all. Also to my wonderful and always- there siblings and their families, Doctor Gbenga Olorunfemi and Barr Tola Olorunfemi, Professor Esther Akinlabi and Dr Stephen Akinlabi, for their sacrifice: to make sure that I was comfortable throughout the duration of my program. Mr Peter Oloba and Mrs Soji Oloba, cousins, nephews, aunt and uncles, Engr Abolade Lawal, Engr and Mrs Gbenga Adebayo, Mary Kayode: you are all sighted and recognized; and all of you are part of my success story.

Also, I want to thank every member and staff of the department of Electrical and Electronic Engineering Science at the University of Johannesburg for their support. I would also like to thank Dr Abiri, Mrs Tshidi Rashama, Mrs Cynthia Oyama (my wonderful roommate), Mr Yinka Abegunde, Mr Damilare Baruwa, Dr Sanusi, Dr Rasheedat Mahamood, Dr Mutiu, Mr and Mrs Shola Olaleye, Mr and Mrs Toyin Ewuola: the list just goes on and on.

Finally, I would like to appreciate all my spiritual guides for their prayers and the love shown towards me and my family during this research work, Pastor and Mrs Raymond Akinsulire (Zonal pastor RCGG victory parish, Suleja, Niger State, Nigeria), all the members of Halleluyah house fellowship, and the good women and children department: I love you all. To my pastors in
South Africa, Pastor and Pastor (Mrs.) Gbenga Ojo (Senior Pastor Pacesetters church, Melvile, South Africa) and all the wonderful members, I thank you all.
GLOSSARY OF TERMS

**Application Program Interface (API):** These are sets of routines and tools used for building software applications. It specifies how the software components should interact with each other.

**Base Transceiver System (BTS):** This is a network tool used to facilitate the wireless communication between the network and the device.

**Intrusion Detection System (IDS):** This can be defined as setting a set of proxy to look for unknown signatures and interpreting them – to know whether the system is under attack, or not.

**International Mobile Equipment Identity (IMEI):** This is a fifteen-to-seventeen digit unique code to identify a mobile-phone set. This enables the network provider to prevent stolen phones from initiating calls.

**Internet Protocol (IP):** The internet protocol provides a standard set of rules for sending and receiving the data over the internet.

**Short Message Service (SMS):** SMS is a 160 alpha-numeric character transmission of short text messages to and from mobile phones.

**Operating System (OS):** This is software that communicates with the hardware in every system, in order to make the programs run.

**Information Communication Technology (ICT):** This is the different combinations of infrastructure to make modern computing easy and effective.

**Denial of Service (DoS):** Denial of service is preventing, hijacking; and hindering the service in a malicious way.

**Global System Mobile (GSM):** This is a digital mobile telephony system used for global-mobile communication.
ABSTRACT

This dissertation is based on developing a mobile application by using an android studio with java programming language to detect incoming unsolicited Short Message Service (SMS) in mobile phones. The Application (APP) was developed and tested on android version 4.4 and 5.0. The results were good; and the App worked as expected. The App works in such a way that whenever there is an incoming SMS from a number which is not in the contact list, there will be room to either ‘allow’ or ‘block’ the incoming message from being delivered. If the user chooses to allow, the message will be delivered; and it would be accessible to the user, otherwise, the message will be blocked and not delivered; and such a contact cannot subsequently send an SMS to the mobile phone. It is believed that this App will enhance the experience of mobile-phones users; because the interface is user-friendly and easy to use.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEDICATION</td>
<td>2</td>
</tr>
<tr>
<td>COPYRIGHT STATEMENT</td>
<td>3</td>
</tr>
<tr>
<td>DECLARATION</td>
<td>4</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>5</td>
</tr>
<tr>
<td>GLOSSARY OF TERMS</td>
<td>7</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>8</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>9</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>13</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>15</td>
</tr>
<tr>
<td>ABBREVIATIONS</td>
<td>16</td>
</tr>
<tr>
<td>CHAPTER ONE</td>
<td>18</td>
</tr>
<tr>
<td>1.1 BACKGROUND</td>
<td>18</td>
</tr>
<tr>
<td>1.2 PROBLEM STATEMENT</td>
<td>19</td>
</tr>
<tr>
<td>1.3 OBJECTIVES OF THE STUDY</td>
<td>19</td>
</tr>
<tr>
<td>1.4 HYPOTHESIS STATEMENT</td>
<td>20</td>
</tr>
<tr>
<td>1.5 RESEARCH METHODOLOGY</td>
<td>20</td>
</tr>
<tr>
<td>1.6 SIGNIFICANCE/MOTIVATION OF THE RESEARCH</td>
<td>20</td>
</tr>
<tr>
<td>1.7 PROJECT LAYOUT</td>
<td>21</td>
</tr>
<tr>
<td>1.8 SUMMARY</td>
<td>21</td>
</tr>
<tr>
<td>CHAPTER TWO</td>
<td>23</td>
</tr>
<tr>
<td>2.1 INTRODUCTION</td>
<td>23</td>
</tr>
<tr>
<td>2.2 DISTINCTIVE INTRUSION-DETECTION SYSTEM FRAMEWORK</td>
<td>23</td>
</tr>
<tr>
<td>2.3 FEATURES OF IDS</td>
<td>24</td>
</tr>
<tr>
<td>2.4 TYPES OF INTRUSION- DETECTION TECHNIQUES</td>
<td>25</td>
</tr>
<tr>
<td>2.4.1 ANOMALY-DETECTION</td>
<td>25</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.4.2 MISUSE-DETECTION</td>
<td>26</td>
</tr>
<tr>
<td>2.4.3 NETWORK- BASED INTRUSION- DETECTION SYSTEMS</td>
<td>27</td>
</tr>
<tr>
<td>2.4.4 HOST- BASED INTRUSION DETECTION</td>
<td>27</td>
</tr>
<tr>
<td>2.5 MOBILE PHONES</td>
<td>28</td>
</tr>
<tr>
<td>2.5.1 TYPES OF MOBILE PHONES</td>
<td>28</td>
</tr>
<tr>
<td>2.5.2 THE CHARACTERISTICS OF MOBILE PHONES</td>
<td>29</td>
</tr>
<tr>
<td>2.6 SECURITY ISSUES ASSOCIATED WITH MOBILE DEVICES</td>
<td>30</td>
</tr>
<tr>
<td>2.7 PHYSICAL MOBILE THREATS</td>
<td>31</td>
</tr>
<tr>
<td>2.8 APPLICATION- BASED MOBILE THREATS</td>
<td>31</td>
</tr>
<tr>
<td>2.9 NETWORK-BASED MOBILE THREATS</td>
<td>32</td>
</tr>
<tr>
<td>2.10 WEB-BASED MOBILE THREATS</td>
<td>33</td>
</tr>
<tr>
<td>2.11 MOBILE VULNERABILITIES</td>
<td>33</td>
</tr>
<tr>
<td>2.12 UNSOLICITED SHORT- MESSAGING SERVICE (SMS)</td>
<td>52</td>
</tr>
<tr>
<td>2.13 ELECTRONIC COMMUNICATIONS AND TRANSACTION (ECT) ACT</td>
<td>52</td>
</tr>
<tr>
<td>2.14 SUMMARY</td>
<td>54</td>
</tr>
<tr>
<td>3.0 CHAPTER THREE</td>
<td>55</td>
</tr>
<tr>
<td>3.1 INTRODUCTION</td>
<td>55</td>
</tr>
<tr>
<td>3.2 OVERVIEW OF UJMS ANTI-VIRUS</td>
<td>55</td>
</tr>
<tr>
<td>3.3 APPLICATION INTERFACE</td>
<td>56</td>
</tr>
<tr>
<td>3.4 ANDROID OPERATING SYSTEM</td>
<td>56</td>
</tr>
<tr>
<td>3.5 ANDROID ARCHITECTURE</td>
<td>65</td>
</tr>
<tr>
<td>3.5.1 LINUX KERNEL LAYER</td>
<td>66</td>
</tr>
<tr>
<td>3.5.2 LIBRARIES</td>
<td>67</td>
</tr>
<tr>
<td>3.5.3 ANDROID RUN-TIME</td>
<td>67</td>
</tr>
<tr>
<td>3.5.4 ANDROID RUN-TIME</td>
<td>68</td>
</tr>
<tr>
<td>3.5.5 APPLICATION FRAMEWORK</td>
<td>68</td>
</tr>
<tr>
<td>3.5.6 APPLICATIONS</td>
<td>69</td>
</tr>
<tr>
<td>3.6 ANDROID’S BASIC-APPLICATION COMPONENTS</td>
<td>70</td>
</tr>
<tr>
<td>3.6.1 Activities</td>
<td>70</td>
</tr>
<tr>
<td>3.6.2 Service</td>
<td>71</td>
</tr>
<tr>
<td>3.6.3 Content providers</td>
<td>72</td>
</tr>
</tbody>
</table>
3.6.4 Broadcast Receiver .............................................................................................................. 73

3.7 ARCHITECTURAL LAYOUT OF THE GLOBAL SYSTEM MOBILE (GSM) IN
RELATION TO THE APP ........................................................................................................... 75

3.8 APP FLOW CHART .......................................................................................................... 76

3.9 UJSMS ANTIVIRUS BLOCKED CHARACTER ALGORITHM ............................................... 78

3.10 UJSMS ANTIVIRUS IMPORTANT CODES .......................................................................... 79

3.10.1 BROADCAST RECEIVER FOR SMS ............................................................................. 79

3.10.2 DELIVERY RECEIPT ...................................................................................................... 79

3.10.3 MAIN RECEIVER PARENT ............................................................................................ 79

3.10.4 MESSAGING RECEIVER MAIN RECEIVER................................................................. 80

3.10.5 GET THE LIST OF ALL THE CONTACT NUMBERS ................................................... 84

3.10.6 READ AND WRITE THE MOBILE PHONE MEMORY .................................................. 85

3.11 SUMMARY ...................................................................................................................... 91

4.0 CHAPTER FOUR ............................................................................................................ 92

4.1 INTRODUCTION ............................................................................................................. 92

4.2 EXPERIMENTAL SET-UP ............................................................................................... 92

4.3 HOW TO LOAD THE APP INTO THE MOBILE PHONE ................................................ 92

4.3.1 Step 1: CONNECTION ..................................................................................................... 92

4.3.2 Step 2: DEVICE SUCCESSFULLY INSTALLED .............................................................. 93

4.3.3 Step 3: DEVICE READY TO USE .................................................................................. 94

4.3.4 Step 4: ENABLE USB DEBUGGING ............................................................................... 95

4.3.5 Step 5: LAUNCH ANDROID STUDIO .......................................................................... 95

4.3.6 Step 6: CLICK THE RUN BUTTON ................................................................................ 96

4.3.7 Step 7: GRADLE BUILD RUNNING .............................................................................. 97

4.3.8 Step 8: APPLICATION SUCCESSFULLY INSTALLED .................................................. 97

4.4 TESTING THE APP ON THE ANDROID- PHONE VERSION 4.4 ................................. 98

4.4.1 MAKING THE APPLICATION TO BE THE DEFAULT SMS ......................................... 99

4.4.2 VIEWING THE SETTINGS AND THE SPAM ............................................................... 99

4.4.3 VIEWING THE SPAM INBOX ...................................................................................... 100

4.4.4 SMS SHOWS IN THE NOTIFICATION BAR ................................................................. 101

4.4.5 ALLOW-OR -BLOCK FEATURE OF THE APPLICATION .......................................... 102

4.4.6 SMS DISPLAYED IN THE INBOX AFTER CHOOSING THE ALLOW OPTION ..... 102
LIST OF FIGURES

Figure 2.1: Intrusion-Detection System (IDS) architecture, adopted from [6] ........................................ 24
Figure 2.2: Anomaly-detection system adopted from [7] ..................................................................... 26
Figure 2.3: Misuse-detection systems adopted from [7] ..................................................................... 27
Figure 2.4: normal flow of information ................................................................................................. 44
Figure 2.5: (a) interruption of information in a system ....................................................................... 44
Figure 2.5: (b) interception of information in a system ....................................................................... 45
Figure 2.5: (c) modification of information of the system ................................................................. 45
Figure 2.6: Malware injections in a mobile phone adapted from [36] .................................................. 51
Figure 3.1: Android 1.6 version adapted from [42] ............................................................................. 58
Figure 3.2: Android 2.1 version adapted from [42] ............................................................................. 59
Figure 3.3: Android 2.2 version adapted from [42] ............................................................................. 59
Figure 3.4: Android 2.3 version adapted from [42] ............................................................................. 60
Figure 3.5: Android 3.0 version adapted from [42] ............................................................................. 61
Figure 3.6: Android 4.0 version adapted from [42] ............................................................................. 61
Figure 3.7: Android 4.1 version adapted from [42] ............................................................................. 62
Figure 3.8: Android 4.4 version adapted from [42] ............................................................................. 63
Figure 3.9: Android 5.0 version adapted from [42] ............................................................................. 63
Figure 3.10: Android 6.0 version adapted from [42] .......................................................................... 64
Figure 3.11: Android 7.0 version (Nougat) ......................................................................................... 64
Figure 3.12: Android Architectural layouts adapted from [43] ........................................................... 66
Figure 3.13: The Linux Kernel layer adapted from [43] ..................................................................... 67
Figure 3.14: the Libraries adapted from [43] ...................................................................................... 67
Figure 3.15: Android Runtime adapted from [43] .............................................................................. 68
Figure 3.16: The application layer adapted from [43] ....................................................................... 69
Figure 3.17: The applications adapted from [43] ................................................................................. 70
Figure 3.18: Android activity ................................................................................................................ 71
Figure 3.19: Android service ............................................................................................................... 72
Figure 3.20: Android Content providers ............................................................................................. 73
Figure 3.21: Broadcast receivers Adapted from [45] .................................................................... 74
Figure 3.22: Android broadcast receiver .............................................................................................. 74
Figure 3.23: Architectural Layout of the GSM system adapted from [47] ....................................... 75
Figure 3.24: UJSMS Antivirus flowcharts ......................................................................................... 76
Figure 3.25: UJSMS antivirus blocked character algorithm .............................................................. 78
Figure 4.1: USB Connection to the laptop ......................................................................................... 93
Figure 4.2: Device successfully installed

Figure 4.3: Enabling USB debugging

Figure 4.4: allowing USB debugging on the laptop

Figure 4.5: Launching the Android studio

Figure 4.6: Allowing the App to run on the phone

Figure 4.7: Gradle building running

Figure 4.8: App successfully installed on the phone

Figure 4.9: App installed on the phone

Figure 4.10: Making the App the default messaging app

Figure 4.11: viewing the spam settings

Figure 4.12: Viewing the spam inbox

Figure 4.13: SMS showing at the notification bar

Figure 4.14: Allow or block features of the App

Figure 4.15: SMS display inbox after allowing

Figure 4.16: Choosing the block option

Figure 4.17: Testing with known contact

Figure 4.18: App successfully installed

Figure 4.19: Theme change option

Figure 4.20: UJSMS default- messaging app

Figure 4.21: viewing the settings

Figure 4.22: view in the spam inbox

Figure 4.23: SMS showing in the notification bar

Figure 4.24: The allowed, or the blocked option

Figure 4.25: Showing the SMS inbox

Figure 4.26: Showing the blocked option

Figure 4.27: Testing the App with a known contact
LIST OF TABLES

Table 2.1: Published literatures on intrusion detection systems in mobile phones. ..................... 34
Table 3.1: History of android versions adapted from [40]............................................................ 55
ABBREVIATIONS

A
APP - Application
API – Application Program Interface

B
BTS- Based-Transceiver System

C
CIA-Confidentiality, Integrity and Authorization

D
DoS-Denial of Service

E
EEG- Electroencephalography

G
GSM - Global System Mobile
GPS – Global Positioning System

H
HMM- Hidden Markov Model
HIDE- Host-Intrusion Detection Engine

I
ICT-Information-Communication Technology
IDS- Intrusion-Detection System
IMEI- International Mobile Equipment Identity
IP- Internet Protocol

ITU-International Telecommunication Union

K
KBTA- Knowledge-Based Temporary Abstraction

M
MANET- Mobile Adhoc Network
MMS - Multi-Media Service

O
OS - Operating System

P
PSTN- Public switched telephone network

R
RBF Radial Basis Function

S
SDN-Software-Defined Networking
SMS- Short Messaging Service
SVM - Support Vector Machine

U
UJ– University of Johannesburg

USB- Universal Serial Bus

V
VFTD- Very Fast Decision Tree
CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND

As telecommunication technology increases everyday, mobile phones users grow tremendously, due to the effectiveness and the number of services they are able to perform thereon, especially when it is connected to the internet. These range from the internet bank transfer of cash from one bank to another for payment of bills, as well as online purchases- to mention but few. According to Jalaluddin et.al, [1], the population of mobile phone users sits at about 6.9 billion around the world, which is equal to 95.5% of the total population of the world, projected in May 2016 by the International Telecommunication Union (ITU) [1].

To lower the infrastructural costs and to appease employees, companies are seeking to employ so-called “Bring Your Own Device” (BYOD) policies. These would allow the workers to gain controlled access to the internal network resources of the company via their personal mobile devices (mainly tablets and phones). These devices typically run a mix of enterprise and personal applications.

Cisco predicts that the number of mobile devices will exceed the world population by 2020 [2].

An inevitable consequence of the huge success of user mobility is the increasing exposure of mobile devices and networks to a wide array of attacks. In addition to eavesdropping on wireless transmissions, break-in GSM masquerading and social engineering; these are all the threats faced by users on the mobile network [2].

Intrusion detection is a security measure that helps to recognize a set of threats that can compromise the Confidentiality, Integrity and Availability (CIA) of the computer network assets of an organization. Assets do not necessarily mean the physical properties of the organization; but this refers to everything that belongs to the organization. This includes the network and the human resources.
Compromising the network of an organization results in an attack on the organization as a whole [3]. Relating this to the mobile devices, malicious software (malware) has come a long way in attacking mobile devices by self-updating and executing Trojan, worm and botnet on the mobile device for their own greedy exploitation.

1.2 PROBLEM STATEMENT

The nature of the mobile-computing environment makes it very vulnerable to adverse malicious attacks. Firstly, the use of wireless links renders the network susceptible to attacks: ranging from passive eavesdropping to actively interfering; since attacks on these links can come from any direction and targeted at any node. This means that a wireless network does not have a clear line of defense; and therefore, every node has to be prepared for encounters with an adversary directly or indirectly. Secondly, mobile nodes are autonomous units that are capable of roaming independently. Since tracking down a particular mobile node in a global scale network cannot be done easily; attacks that compromised any node from within the network are more damaging; and they are harder to detect. Thirdly, decision-making in the mobile computing environment is sometimes decentralized; and some wireless network algorithms rely on the co-operative participation of all the nodes and the infrastructure [4]. Furthermore, mobile computing has introduced new types of computational and communication activities that seldom appear in a fixed or wired environment.

In this respect, relating this to a mobile phone, which is vulnerable due to its features of open medium, portable device, easily attacked, lack of centralized monitoring and management point, and no clear line of defense; hence, the need for an intrusion-detection system in such devices is of paramount importance.

1.3 OBJECTIVES OF THE STUDY

The premise behind intrusion-detection systems involves deploying a set of proxies to inspect network traffic and look for the “signatures” of known network attacks. This is essentially what an Intrusion Detection System (IDS) does. An (IDS) does not usually take preventive measures when an attack is detected; it is reactive rather than being a pro-active agent; and it plays the role of an informant, rather than that of a “police officer”. In relating this to mobile devices, there are many types of attacks that can occur on mobile devices, caused by malware - ranging from theft
of the data, denial of service (Dos) (i.e. crashing of the phone), stealing users’ identity and theft of resources to mention; but the few are the main aspect of this research work. The main objective of this study is to develop a mobile application by using an android studio with Java programming language to detect incoming unsolicited Short Messaging Services (SMSs) of numbers not on the contact list of the mobile and get it blocked if not wanted by the user. As part of the features of the App, there will be an interface where mobile-phone users can either allow or block the SMS. When the user decides to choose the allow button; the SMS goes directly into the mobile phone inbox messaging. On the other hand if the user decides to choose the block option, the SMS get block permanently and any subsequent one will not come through to the user’s mobile phone.

1.4 HYPOTHESIS STATEMENT

A mobile application is expected to be developed using the android studio with Java programming language-together with the supporting ability to be installed and used on android 4.4 and subsequent versions.

1.5 RESEARCH METHODOLOGY

The target version will be the minimum of (Application Program Interface) API 19, and the maximum of API 25; because it covers a lot of the population of about 1 billion users. The application will be installed on the targeted versions of android-mobile phones for testing. It is expected that when an unsolicited SMS or spam SMS is received, the App will prompt the user to grant permission for the SMS or to be blocked. If permission is granted; the SMS would be displayed in the mobile phone-message inbox; but if it is blocked; it gets deleted permanently, thereby preventing any malicious attack on the mobile phone through unwanted and spam SMS; and thereby improving the performance of the mobile device.

1.6 SIGNIFICANCE/MOTIVATION OF THE RESEARCH

Mobile phone technology is a dynamic technology that keeps evolving constantly, with the development of new Apps and improving the existing ones. Overtime, users have experienced tremendous unprecedented development in mobile-phone technology. The original motive for the creation of the mobile phone was for making calls and sending SMSs; but development of the
technology has changed the original scope; and it has led to the development of sophisticated mobile phones that have the ability of doing sophisticated functions like internet banking, GPS navigation, social-media networking etc. With these additional functions, and developments, it is imperative to also create a better security platform that enables users to experience uninterrupted services; and to protect them from malicious attacks from cyber criminals or hackers. People these days store much of their personal information in their mobile phones, such as credit card numbers and Pins, as well as having access to their emails through the phone etc. Therefore, it has become necessary to protect and keep such informations secure from malicious attacks, and safe from intrusion, or any other interference when communicating with one another, that is why it is necessary to conduct research into that aspect of intrusion in mobile phones, such as unsolicited SMSs.

This research works should lead to article publications; and contribute to the collection of knowledge.

1.7 PROJECT LAYOUT

This dissertation comprises five chapters; Chapter One gives a brief introduction, which comprises: the Background, the Problem statement, the Objectives of the research, Hypotheses, as well as the Significance of the work. This followed by Chapter Two, which is a critical review of the related works; types of intrusion-detection systems, security, threats and the vulnerabilities of mobile devices, as well as the different kinds of methods and designs, which have been used before in intrusion-detection system (IDS) both in wired networks and mobile devices and their outcomes. Chapter Three gives elaborate writing, the research methods, the coding and the flow charts. Chapter Four focuses on the experimental set-up and the testing of the App. Finally, Chapters five draws the conclusion; and it suggest some remarks.

1.8 SUMMARY

Conclusively, in this chapter, it is important to conduct research into the intrusion and detection of unwanted SMSs in mobile phones. This chapter comprises sections, like the background of the work, the problem statement, the objective of the research work, the hypothesis statements, the research methodology, the significance/motivation of the project; and finally, it gives the lay-out
of the project. The next chapter will present a critical literature review of some similar work that have been done before now.
CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

Intrusion-detection systems have come a long way, this came to light because of the rate at which attacks and vulnerabilities have grown increasingly over the network, especially the mobile network. However, the growth in the market for mobile devices like Smartphone, tablets and laptops etc, has greatly increased because of the luxury features that these devices present to the users, when connected to the wireless networks, which allows it to be carried from one place to another. Intrusion detection involves capturing the audit data and interpreting the evidence in the data to determine whether the system is under attack or not. A computer system is expected to provide Confidentiality, Integrity and Authorization (CIA) to the user. However, due to increased connectivity (especially on the internet), the vast spectrum of financial possibilities that are opening up, more systems are subject to attack by intruders. These subversive attempts try to exploit the flaws in the operating system, as well as in the application programs; and they have resulted in spectacular incidents such as the Internet Worm incident of 1988 [5]. Robert Morris invented the coded red worm in 1988, which affected about 6000 personal computers on the network, and also the first denial of attack, which was first launched in 2000 on the computer systems - including large companies, like Yahoo, eBay, Amazon and CNN. All these major attacks contributed to the invention of the intrusion detection systems. In this chapter, the related theoretical background will be explored and discussed in detail; the published literature in the field of IDS: both in the wired network and mobile phones will be critically reviewed and presented.

2.2 DISTINCTIVE INTRUSION-DETECTION SYSTEM FRAMEWORK
The diagram presented here represents an intrusion detection framework, from the sensor to the data collector; from there; it generates the profile; and then it moves on to the IDS itself; where it analyzes and sends to the post processor, which updates the profile; and finally it presents the profile templates, as the final result.

![Diagram of Intrusion-Detection System (IDS) architecture](image)

Figure 2.1: Intrusion-Detection System (IDS) architecture, adopted from [6]

### 2.3 FEATURES OF IDS

Typically, an intrusion-detection system consists of the following features:

- **Data collector**: This collects the important data from the sensor on the systems.

- **Profile generator**: The profile generator analyses the data from the data collector; and it then generates the profiles.

- **Profile templates**: This stores up the profiles analysed from the profile-generator; meanwhile, the signature based IDS saves malicious profiles, such as (signatures); while at the same, time an anomaly-based IDS saves the typical profiles.

- **The intrusion detector**: The intrusion detector is the main component in the IDS. This performs the detection of the intrusion, using the most recent profiles that have been generated.
**Post processor:** The postprocessor is the one in charge, who takes proper actions against the intrusion; once it has been detected [6].

**Profile updater:** This is in charge of taking complete and accurate updates by using the most recent profiles and important algorithms.

### 2.4 TYPES OF INTRUSION-DETECTION TECHNIQUES

There are four major types of intrusion-detection techniques; and they are here highlighted and discussed:

#### 2.4.1 ANOMALY-DETECTION

The anomaly-detection method assumes that all the interfering behaviour is essentially anomalous. This means that if we could create a "normal activity profile" for a system, we could, in theory, flag all system states - varying from the well-known profile - by statistically significant amounts as intrusion may want to occur. However, if we consider that the set of intrusive activities only intersects the set of anomalous activities, instead of being exactly the same; we find a couple of interesting possibilities:

1. Anomalous behaviours that are not intrusive are flagged as intrusive.

2. Intrusive behaviours that are not anomalous end up as in false negatives (actions are not flagged as been intrusive, even though they are). This is a dangerous problem; and it is far more serious than the problem of false positives.

Anomaly-detection systems are computationally expensive, because of the overhead of keeping track of, and possibly updating several system-profile metrics. A block diagram of a typical anomaly-detection system is shown in Figure2.2 adopted from Madhavi and Kim [7].
2.4.2 MISUSE-DETECTION

The idea following misuse detection system is that there are ways to represent attacks in the form of a pattern, or a signature, so that even variations of the same attack can be detected. This means that these systems are not different from virus-detection systems; since they can detect almost all well-known attack patterns; but they are of little use for the detection of as-yet unknown attack methods. An interesting point to note is that anomaly-detection systems try to detect the complement of "bad" behaviours. Misuse-detection systems try to recognize known "bad" behaviours. The main issues in misuse-detection systems are how to write a signature that encompasses all possible variations of the pertinent attack; and how to write signatures that do not also match non-intrusive activities. A typical diagram of a misuse-detection system is shown in Figure 2.3.
2.4.3 NETWORK- BASED INTRUSION- DETECTION SYSTEMS

Network-based intrusion detection is performed over the network data, such as network traffic, data packets etc. Network-based intrusion detectors are inserted in the network-just like any other device; to examine every packet they observe on the network, in order to detect intrusions. This method has its own advantages and disadvantages. Advantages, such as reducing the processing overwork from the mobile phone, detecting external intrusions etc. Disadvantages, there will be no access to monitor the data on the mobile phone that could be useful for detection. Communication setting is very disjointed; as a mobile device can be connected to many sources on multiple interfaces at the same time; and it cannot detect intrusions to the device, such as malware etc.

2.4.4 HOST- BASED INTRUSION DETECTION

While network-based intrusion detectors are straightforward to deploy and maintained; there is a whole class of attacks closely coupled to the target system; and which are extremely difficult to fingerprint. These are the ones that exploit vulnerabilities particular to specific operating systems and application sites. Host-based intrusion detections are performed, based on the information available on the mobile phone, such as central processing unit activity, memory consumption, file input and output activity, network input and output activity, operating system level events, application interface level and measurements etc. It also has its advantages and disadvantages. The advantages are having access to the private information on the mobile device that is of use to
detect intrusions; as the information is collected together from the mobile device. This would reveal the device-performance correctly; and intrusion-detection models with host-based data collected works give more exact and reliable outcome than other approaches etc. This detection system also comes with its own disadvantages, such as difficulty in implementation because of the processing limitations of mobile phones and complexity in providing security for the data that is directly collected from the mobile device [8].

2.5 MOBILE PHONES

Mobile phones are often referred to as cell-phones or cellular phones. A mobile phone is an electronic communication portable device that can make calls and also receive calls over the radio-waves connection, or satellite- transmission, provided the user is within a telephony service area. The radio frequency connection establishes a link between the switch system of a mobile phone and the mobile- phone operator, thereby providing access to the public switched telephone network (PSTN). The current mobile telephony services use the cellular network architecture for their configuration, therefore, mobile phones are also being referred to as cellular telephones, or simply as cell phones. Mobile-phone technology is a dynamic technology that keeps evolving constantly, with the development of new Apps and improvements to the existing ones. Overtime, we have experienced tremendous unprecedented development in mobile-phone technology, which makes the mobile phones to become more sophisticated; and it has capability of doing things, like sending emails; internet banking; online purchasing; GPS navigation, video games, digital photography and social-media networking [9].

2.5.1 TYPES OF MOBILE PHONES

Mobile phones have evolved significantly over time; as their use and the demand for them is high; because of the latest trends they present to users. Mobile phones can largely be classified into three categories, based on their features. These are standard mobile phones; middle-ware standard mobile phones, and new standard mobile phones.

a. **Standard mobile phones**: This type of mobile phones is often referred to as the basic or conventional phones; and these have the basic features in phones, as like making calls
and send text message. It has no advanced features, instead manufacturers have focused on the easy-to-use for users, with big keyboard for texting and typing, attractive flip design and they also come with low price tags.

b. **Smartphones:** Smartphones have become the preferred phone for most mobile phone users. They come with a strong operating system, a good and strong third-party application on them, fast internet-processing speeds and (Global Positioning System) GPS etc. This type of mobile phones has advanced more than the standard phones which is basically for making calls, it has highly advanced features, such as high response touch screen, digital camera, WI-FI connection capability and video-conferencing etc. A Smartphone can perform nearly all the jobs that a computer can do. It performs jobs that were formally meant for laptops or desktops, such as editing word, social Medias, read and write emails; and it can even create spreadsheet. The four popular operating systems found on this device are Apple, Android, windows and Blackberry.

c. **Middle-ware standard mobile phones:** The middle-ware mobile phone is totally different from the Smartphone; and also less expensive. It has a limited proprietary operating system when compared to the strong and robust operating system found in Smartphones. Also middle-ware mobile phones do not support third-party applications; as seen in Smartphones. The features are basically similar to Smartphones; but they still have a lot of differences, such as problems with calendar synchronization, spreadsheet and document editing etc. In this new context, the perspective is on the focused areas; on the multimedia, web browsing, texting and GPS.

Manufacturers and developers make sure they put on their best at producing the best and most trending mobile phones to the marketplace, in order to suit mobile-phone users. However, users go for the ones that suit their needs at a particular-time not minding the cost implications that might come with such sophisticated mobile phones [10].

### 2.5.2 THE CHARACTERISTICS OF MOBILE PHONES

The features and descriptions of mobile phones are hereby presented:

a. **Mobility:** This is the most essential feature of mobile phones, since it is a durable hand-held device, which users can take from one place to another wherever they go. This
exposes them to attacks of being stolen, getting lost and getting tampered with, when compared with the stationary ones.

b. **Strong personalization:** As a personal mobile device, phones are not usually shared among multiple users; and they are meant for individual use.

c. **Strong connectivity:** Mobile phones are generally used to connect to each other, or to other devices over the internet by using a wireless-data connection.

d. **Technology convergence:** Several technological advances are embedded in the Smartphones, such as gaming, video conferencing, chatting and use of the social media etc [11].

A mobile device also has four major limitations, compared with a stationary phone. These are presented below:

a. **Limited battery time:** The life span of the battery of a mobile device is limited; because it is being carried from one place to another, without being constantly charged compared with the stationary ones.

b. **Limited computing power:** The stationary device has the capability to process data and present information faster than the mobile ones.

c. **Small display screen:** The mobile phones have a smaller display unit than that of stationary ones.

d. **Small sizes of keys:** The keys for the mobile device are relatively small compared with the stationary ones in terms of input into the system [12].

### 2.6 SECURITY ISSUES ASSOCIATED WITH MOBILE DEVICES

In a wired network, there are a lot of protections, safe guards, rules and regulations, which prevent an intruder and make the communication secure through devices like firewalls, secured gateway etc. - unlike the mobile network with several decentralized and dynamic nodes, which connect the frequency with the computer for humans to use. However, these make the network more vulnerable to various forms of attacks.

It is, therefore, safe to know these security vulnerabilities and attacks, in order to guard against them and also defensive mechanisms to overcome them. This is the main focus of this research work (IDS). Nisha et.al. [13] highlighted some problems faced by mobile phone users; these are captured as follows:

a. **Unsafe data storage:** The personal information and data, such as credit card numbers, email correspondence, SMS messages and more than one stores on the mobile device. They are totally unsafe if the device is lost or stolen. The thief can also steal the sensitive information in the device.

b. **Inappropriate session-logging management:** Session login management is very important in both mobile devices and PCs, especially when surfing the websites
on the internet; and long-session login is very dangerous; since hijackers can use such vulnerabilities, and login to one mobile device and steal valuable information. As such, session-login management should be brief; and when it’s taking longer, the session should be logged out automatically.

c. **Several logins:** Users with one universal password for their social networking accounts and other confidential accounts, such as emails, bank transfer etc, can easily be hacked into; and they are prone to cyber-attacks. As such, this should be discouraged among mobile-device users.

d. **Weak validation:** Conventional passwords have been the most commonly used among mobile users; but this is not good enough to secure the mobile devices. Alongside the password; there must be another level of security combined with the password; just as [13] proposed face recognition for mobile-device authentication.

The security problems encountered by mobile phones users are grouped into two categories, according to the various types of attack they may exploit on the system. Basically, we have mobile threats and vulnerabilities. Mobile threats can be further divided into four sub-headings, namely: Physical threats; Application- based threats; Network-based threats; and Web-based threats.

These threats will now be discussed.

### 2.7 PHYSICAL MOBILE THREATS

A physical threat is one of the major concerns faced by the mobile-device users; today because it’s a type of device that is portable; and it can be carried from one place to another. Under the physical threat we have, the theft of mobile phones, or their loss as well as the Bluetooth problem [13].

a. **Theft of mobile device or loss:** The loss of mobile devices is a major threat to mobile-device users; because the device can be resold in the market and one’s personal data are then lost, which can threaten user-private sensitive information.

b. **Bluetooth problem:** Bluetooth technology is a short-range frequency wireless connectivity that allows two devices that are paired with each other to connect. The default password allows the transfer of malware from one device to another through Bluetooth, which poses a great risk to the devices.

### 2.8 APPLICATION- BASED MOBILE THREATS

a. **Trojan Threat:** Trojan is a major mobile malware, which utilizes the techniques of social engineering by downloading itself on mobile phones - without the user’s approval. It makes changes to one’s phone by sending unsolicited SMSs; it increases the phone
tariff and allows attackers to have full access to the system by exploiting their vulnerabilities [14].

b. Privacy Threat: This is a kind of threat, which maliciously reveals the International Mobile Equipment Identity (IMEI) Number and device-location information of mobile phones. This results in successful attacks on mobile phones; and it violates the user’s privacy. The various types of privacy leakage include: the IMEI number, the device location, the phone number, Call logs and SMS logs.

c. Adware applications: These sets of applications creep into the phones deceptively; and they install themselves without the user’s permission. These are very difficult to identify or to get rid of from the system. The purpose of these applications is to show off unwanted advertisements to users; but it also serves as a perilous spyware that watches the activities of the users; and it steals private information such as call logs, email, Short messages (SMS), photos and financial data, which can be used as a theft of identity against the user [14].

d. Session-Initiation Protocol (SIP) Threats: The session initiation protocol is a connection procedure for signaling and controlling multimedia communication connections, such as video and voice calls. The main use of SIP is for internet calls and instant messaging over the internet protocol (IP) networks. Vulnerabilities in this SIP are seen as major windows for attackers to exploit the mobile devices, which can pave the way for attackers to launch an attack against the devices, by listening to confidential voice communications, as well as launching denial of a service attack against the mobile devices; and this sometimes crashes the devices [15].

2.9 NETWORK-BASED MOBILE THREATS

Cellular networks and wireless give full support to mobile devices for them to operate; while these in turn, have their own threats and vulnerabilities presented to the mobile-device users through the network. Some of these threats are here highlighted:

a. Cellular or Wireless Network Service: Cellular or wireless network services, such as Multi-Media Messaging (MMS), Short Message Services (SMS), video calls and voice calls can be the source of attack on mobile phones. A phishing attack can occur through these, such as stealing private information from the user by pretending to be an authentic program to the users.

b. Wi-Fi Tapping: This is the interception of information that passes between the mobile device and the Wi-Fi access point through the space. Therefore, it is an immense risk to pass unencrypted information through the Wi-Fi; because it can easily be hijacked.

c. Denial of Service (DoS) Threat: Do’s is an attack or malicious program that denies one to have full access to a particular service or application, as well as other services on the
system. In the case of mobile devices, like Smart phones, it forcefully takes over the mobile device and renders it insecure.

d. **Network-Exploiting threats**: This type of threat exploits the flaws in the mobile device’s operating system and other software applications that operate on either the cellular or wireless network. Subsequently, when the mobile device connects to the network; malicious programs have been installed on the software applications- without the prior knowledge of the user [16].

### 2.10 WEB-BASED MOBILE THREATS

Web-based mobile threats occur when a user attempts to access some web browser over the internet; and as such, it opens chances to threats and vulnerabilities to exploit the mobile device. Some of the web-based threats are described below:

a. **Phishing-trick threat**: phishing is a way of getting useful and valuable private or business data from the user; using links like the social media networks, or emails on a malicious website. The purpose is to steal valuable information, or trying to gain access to a user’s log in details.

b. **Drive by downloads threat**: This is a phenomenon, whereby a user visits a malicious website. The application automatically downloads malicious applications on the device, causing it to be ineffective. This can be prevented by always being careful of the type of website one surfs and not click on any unsolicited downloads.

c. **Browser-exploit threats**: This type of threat takes advantage of the flaws in the mobile phone browser application, such as music files, flash share, PDF files and Gallery. It attacks when surfing on a malicious browser, which tends to steal user-sensitive personal information [16].

### 2.11 MOBILE VULNERABILITIES

Mobile vulnerability is a breach in security system of a mobile device, due to flaws or weaknesses which occurs by the developers during the developing stages in mobile devices. The vulnerabilities include:

a. **Root kit vulnerability**: generally, root kit affects the operating system of the device by installing malicious application on the device, its purpose is to intrude the security of the device and steal information.

b. **Worm vulnerability**: worm is a malicious code that duplicates itself on the mobile device through the network. A worm compromises the defence of a mobile device.

c. **Trojan horse vulnerability**: A Trojan horse inflicts the device with malicious application, such as worm, in order to steal valuable data, which it does through phishing scams.
Some of the literatures reviewed are hereby summarized in the table 2.1.

<table>
<thead>
<tr>
<th>Names of Author</th>
<th>Year of Publications</th>
<th>Intrusion detection systems used</th>
<th>Detections in mobile phones</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azzedine and Mirela</td>
<td>2002</td>
<td>Neural – networks and pattern recognition</td>
<td>Fraud</td>
</tr>
<tr>
<td>Grant et.al</td>
<td>2004</td>
<td>Battery- Based IDS and Host Intrusion Detection Engine</td>
<td>Attacks on battery source</td>
</tr>
<tr>
<td>Yap and Ewe</td>
<td>2005</td>
<td>Misuse detection on behaviour checker</td>
<td>Malware application (Trojan)</td>
</tr>
<tr>
<td>Michalopoulos and Clarke</td>
<td>2006</td>
<td>Host – based, statistical model and Biometrics</td>
<td>Fraud and abnormal activities on mobile phone</td>
</tr>
<tr>
<td>Muhammed and Abdulhamit</td>
<td>2010</td>
<td>Host – based and AMOXID</td>
<td>Threats</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Threat to user experience</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Privacy infringing</td>
</tr>
<tr>
<td>Rohit and Thangakumar</td>
<td>2012</td>
<td>Cloud based and recovery system for Android</td>
<td>Misbehaviours in network</td>
</tr>
<tr>
<td>Donny et.al</td>
<td>2013</td>
<td>Biometrics and Dongle technology</td>
<td>Theft</td>
</tr>
</tbody>
</table>
The highlights of these papers are hereby presented:

Behavior-based intrusion detection using artificial neural network to detect fraud in mobile phone systems. The mobile phone has taken the centre stage in communicating and transacting in this dispensation. Therefore, there is need to provide an online-security system, in order to protect both the device and the users. Azzedine and Mirela [17] developed an online-security system to detect fraud when taking calls. Their work was based on an artificial-neural network with Radial Basis Function (RBF). The RBF include models that can recognize patterns and classify them accordingly. They were used to take past history and recent information about the activities and the usage of the mobile phone. The users are classified into groups, such as where to call, what time to call, from where the calls were made and network used etc., days of the weeks, international or local calls - and then save them in a file, in order to use them as checker, whenever a fraudster, or a genuine user is calling. The security solution was also used for both security systems against cellular-phone cloning and as a system to identify probable imposters and perpetrators. The system works in such a way that if a user is calling, it will be checked
against the saved log files; also whether it correlates with what they have in the saved file. The call goes through unless it is denied. Then both the genuine user and the service provider would be notified immediately. This has helped the telecommunication company in reducing their profit losses; and also the mobile phone user; since it reduces the cost of making calls [17].

Battery-based intrusion detection using a host-intrusion detection engine to detect attacks through battery consumption. Power consumption in hand-held devices is very critical to the usage of the devices. If there is was no power, there would be no computing for hand-held devices. The normal life expectancy of a mobile-phone battery is at least 30 days. If there are less than this – then it emits signals that the device maybe under one or more attacks. Therefore, battery power is very essential in mobile devices. This aspect needs to be researched, in order to detect any attacks draining the power of the battery; and also to protect the devices. However, Grant et al., [18] in their paper entitled “Battery-Based Intrusion Detection” designed a system called Host-Intrusion Detection Engine (HIDE). This monitors the power behaviour to detect possible intrusion by taking into consideration the irregularities of the power consumption in the device. This is done by measuring the consumption of the power over a period of time (T) and energy (E) to determine whether the device is under an attack. The system also provides an early alarm, in order to detect any attacks - before irreparable damage is caused to the device; and it also protects the battery lifecycle [18].

A mobile-phone malicious software detection model using misuse detection as a Behaviour Checker. Mobile-phone users need to get a good security system, in order to be free of various types of malicious programs finding their way into the applications that run on the device. Mobile malware, such as Trojan, virus and worms are on the rampage, in order to steal user information that is stored on the phone by always sending SMSs to users, without the user’s acknowledgement, which would bring high financial costs to the users. Consequently, Yap and Ewe [2005] in their work entitled “A Mobile Phone Malicious Software Detection Model with Behavior Checker” have come up with a model, which can be used to detect malware in mobile devices and to be specific (Trojan malware). They used a misuse-detection system, based on a behaviour checker.
Misuse detection is very efficient and fast at detecting the malicious behaviours of known attacks. Since a mobile phone has only limited features, it was easy to compile the database of behaviours known attacks. Furthermore, it can detect malicious applications which tend to send SMSs - without notifying the mobile-phone users; while the behaviour checker monitors the activities of the applications that runs in the system.

The proposed solution works in such a way that whenever an application wants to use any of the messaging features, it needs to make a session with the message server. A detector application will now observe all the sessions and monitor all the messaging events in the session. If it correlates with the known attack, it is suspended; and it notifies the user, as follows: “ACTIVATE_MONITOR_RESPONSE_SUSPEND LIST”

The app was tested on a Nokia mobile phone with a symbain operating system; and it works as expected [19].

Intrusion detection in mobile phones can be done by using host – based, statistical model and Biometrics. The number of mobile-phone users has increased drastically in recent years - due to the features that now come with them. The amount of information stored on these devices is considerable; and the security can no longer be compromised. Therefore, the single authentication and validation is no longer adequate for mobile-phone security. Michalopoulos and Clarke [20] in their paper entitled: “Intrusion-Detection System for mobile devices” they developed a host based IDS system, based on a statistical model and Biometrics in 2006, in order to detect fraud and abnormal activities on mobile phones. The system algorithm was developed with multiple sensors installed on the device to monitor user altitudes and keep tabs of the records of the day-to-day activities. The system consists of two layers. The first layer is to give a warning to the user that an abnormal behaviour is suspected. It is then necessary that the user re-authenticates itself, in order to be able to continue with the device. The second layer consists of an alarm-triggering sensor. If the user ignored the first warning, the alarm was triggered, thereby notifying the user that another authentication is required for further authentication; and this time it must be biometric, in order to continue with the use of the device. The system was developed and tested and it works as expected which improves the security of the mobile phones when coupled with the mobile-phone user’s experience [20].
Intrusion detection using host-based approach and policies to detect threats in Smartphones, in addition to Smartphones, technologies are increasing every day; and this makes the users also to increase - due to the sophisticated features they offer to the users. Smartphone users can do a lot on their phone, such as checking emails, formatting word documents, excel; and they can do online banking and online shopping etc. If they are not well-protected against threats and malware, the confidential user PIN could be hacked into and used to steal their money. Therefore, Muhamed and Abdulhamit [21] proposed a model “AMOXID” to cope with this problem. This has different network policies and guidelines, to which the users must adhere when surfing a particular network. They categorized the Smartphone threats into three broad headings, namely:

a) Threat of Smartphone user experience: this kind of threat is associated largely with the device resources, such as the threat of draining the phone battery and the memory. When the phone battery is under constant attacks by threat, it makes the users worry about the constant charging of the phone; while attacks on the memory slow down the processing speed of the phone; and this renders it unresponsive; and makes the user’s experience unsatisfactory.

b) Threat of cost loss: this kind of threats is associated with loss of money the potential hackers get from outgoing call, outgoing SMS etc.

c) Threat of privacy infringing: privacy infringing deals with stealing confidential data and information that are stored on the mobile phone such as the phone book, the contacts, user location etc.

They categorized the networks into three headings and provided the policies to be used for each network, in order to be safe from threats. Also the network manager must enforce and make sure the policies are being adhered to properly. The alert manager get users alerted when any new threats are found. These are hereby presented:

a) Home network: these are networks normally used in the house; and they are private and confidential.

b) Work network: this type of network is usually used in an office environment, and is properly managed.
A cloud-based intrusion-detection system for Android smartphones, cloud-based and recovery system for Android was used to detect any misbehaviour in a network. Cloud computing is basically used, in order to get rid of the local information storage. The smartphones being produced these days comes with advanced processors, operating systems; and they are able to perform work that a computer can do successfully. This makes the phones vulnerable to various attacks and malicious software applications, such as a virus, Trojan and worms. Against this background, a study was conducted by Rohit and Thangakumar [22] in their paper entitled: “A Cloud-Based Intrusion Detection System for Android Smartphones” in which cloud computing services were used as the intrusion detection and recovery process. The cloud service was deployed with a mobile-host agent, known as lightweight, which was installed in the form of an application on the android-mobile device. The system uses a proxy, which duplicates the exact nature of the device; and it is stored in the cloud. It also duplicates all the traffic between the mobile device and the internet; and it then send them to the cloud environment for an in-depth inspection to be carried out on them; while the cloud-computing service takes input from the mobile device and performs the detection and identification of any malicious abnormal behavior over the network. It also serves as the access control for the mobile device, as well as any applications and the services in the device. The lightweight does the recovery where necessary. The architecture has four basic components, which are: the emulator, which copies the exact residence of the mobile device in the cloud; the memory scanners: they scan the device for resident attacks, in order to act against such attacks. The system was called the anomaly detection: and the system calls the detection of an anomaly very fast. Therefore it is applied solely as the system calling device; and lastly, the antivirus software: this runs in the emulator to detect the abnormal malicious signatures, such as Trojan, viruses and worms. The system proved to be good in detecting malicious applications; and it also serves as recovery process whenever necessary [22].

Fingerprint biometric security is accomplished by utilizing dongle solid-state relay technology to detect theft in mobile phones. Mobile phones are usually open to attack; and the threat, especially that of theft when left unattended in public places. In protecting these devices, the
focus should not be only on the information and data in the mobile phone, but also on the mobile phone itself. The use of a conventional password and drawing patterns are no longer enough to secure the mobile devices; therefore, there is a need for more comprehensive security solutions. The use of biometrics has also been deployed in several approaches, such as voice, face and fingerprint. Unfortunately, these biometrics alone are not good enough to secure the device. Consequently, Donny et.al. [23] proposed a biometric approach combined with the biometric charger, which serves as the device dongle. The mobile phone and the mobile-phone charger will be utilized as capacitive fingerprint readers, which enables the full functionality. The system proposed that when a mobile phone is bought, it should be programmed with the user fingerprint; as well as the mobile-phone charger, which is also programmed with the user fingerprint, and this should be done in a way that it can only be the mobile phone manufacturer that can do the re-programming, if required. “The fingerprints then become an encrypted key, which the two devices need to use for synchronization. The battery is recommended to be in-built and should not be detachable from the phone, such that if separated from the device, the mobile phone becomes useless; because both devices need to synchronize the fingerprints before it can work. If this solution is deployed; the theft of intrusion of mobile phones would be discouraged” [23].

A multi-agent system-intrusion detection is used to detect SMS-botnet in android Smartphones. Mobile SMS botnets have been increasing greatly; as the number of Smartphone user’s increases. SMS botnet is a dangerous malware in Smartphones that causes users more harm than good. They exploit Command and Control (C and C) channels to send a spam SMS and premium-rate SMS to users’ phones without prior user knowledge. In light of this, Abdullah and Ali [24] in their work entitled “A Multi-agent System for Smartphone Intrusion Detection Framework” they built a framework using a Multi-agent system design and hybrid-based IDS to detect SMS-botnets in android Smartphones. The framework has the capability to check and examine Smartphone activities; it captures the logarithmic functions and sends a report of a suspected anomalous behaviour to the central server to analyze. Their framework consists of two broad components, namely: the multi-agent system and the Intrusion-Detection system. The multi-agent system has two sub-division under it, which perform some specific tasks and roles, in order to make the system complete:

1. The android-mobile device and the central server.
2. Intrusion-detection system

The android mobile device has four agents - all working together to achieve the same purpose, namely:

a) The manager agent: this creates a communication channel between the Smartphone and the central server.

b) An SMS-detection agent: who monitors both the incoming and outgoing SMSs, it also sends a request to the signature-based intrusion system to detect any anomalous SMSs found.

c) Monitoring agent: this is responsible for observing the phone activities, the phone settings; and it also monitors the phone; while remaining connected to the internet.

d) A human-behaviour agent who monitors the mobile-phone user’s connectivity time, reports daily phone usage; and it responds to the agent manager, as required.

The central server consists of three agents, namely:

a) The central agent: this has the responsibility of responding to the phone by adding any new device to the subscribing list, updating, blocking and deleting as required.

b) SMS profiling agent with a function to examine suspected SMSs and send to the detection module to analyze; its major function is to interact with the detection module.

c) Android-profiling agent: this does continuous monitoring of the profile databases for all subscribers; and it updates any changes, as required.

The intrusion-detection system, consists of three action modules, namely:

a) Signature-based detection module: this performs the task of scanning through for any known signature that is malicious; and it sends this to the next module to analyze and perform the required action, as the case may be.

b) Detection module: the detection module performs the collection of SMSs; and it classifies the SMSs. It also does the major part of the work in detecting malicious SMSs and URLs sent from the detection module.

c) Decision-and-action module: this is the last module of this segment of the framework. It uses the results and the information from the detection module to analyze and classify the
phone numbers of the malicious SMSs and URLs; and it blocks them, as and when appropriate.

All these were integrated together and able to detect the botnet SMS sending to user mobile devices through SMS without the user’s prior knowledge; and it was very efficient without exhausting the mobile-phone resources, such as the memory and the battery etc. [24].

Intrusion detection in mobile phones using Bluetooth Logging Agent (BLA). Mobile phones with Bluetooth-enabled features exposes the devices to great risk. The sensitive information on the mobile phone can easily be stolen - if there are no proper security mechanisms in place for such mobile phones. Kishor et.al. [25] in their work “Intrusion Detection in Bluetooth- Enabled Mobile Phones” developed a system called Bluetooth Logging Agent (BLA), in order to detect malicious and unauthorized access in wireless network through Bluetooth features of the mobile phone, by either accepting or rejecting any connection that the user chooses. This works with another system called Bluetooth mobile-module logging which houses the databases built; and it uses verification rules and authentication to verify all the communications between the mobile phones and the remote devices. When a communication is initiated, an alarm system was put in place that notifies the mobile phone users that there is an intruder. The system was tested; and it was discovered that it improves the authentication of the Bluetooth-enabled device security [25].

“A bio-signal based framework to secure mobile devices using Electroencephalography (EEG) with Statistical analysis, Hidden Markov Model (HMM) and Support Vector Machine (SVM) to detect unauthorized access”. The EEG technology is a new birth of technology that is used in medicine to diagnose patients brain disorders, migraine etc. since brain waves represent the physiological and behavioral information about a person. It has also come into the security field; it is now used to detect unauthorized users, which know the genuine user-draw pattern and want to intrude into the mobile device. The work developed an authentication system for mobile devices using Electroencephalography (EEG) with wireless connectivity through Bluetooth means with the Headset-Hidden Markov Model (HMM) and the Support-Vector Machine (SVM). The signals of fifty (50) different users of their brain waves were taken; twenty (20) were of genuine users and thirty (30) of unauthorized users, when drawing their various
personalized unlocking patterns on their mobile devices. Signals of users are simultaneously recorded to classify a genuine user from forgers. The brain signals were captured by using EEG android APIs Smartphones connected to the EEG headset by using Bluetooth technology. Consequently, the system experimented with the three standard bio-metric security metrics, namely: Half Total Error Rate (HTER), Receiver Operating Characteristics (ROC) and Detention Error Trade-off (DET). When a forger is trying to get unauthorized access to the user’s mobile device by drawing the user pattern, which the forger must have stolen from the user, the EEG receives the forger’s brain waves and compare it with that of the genuine user. If it matches, is authenticated; otherwise the access is denied. The work was tested and result was efficient; and it was a plus to the security of the mobile devices [26].

Furthermore, according to Marti et.al. [27], any intrusion into a system can be regarded as a threat. Threats can be seen as potential violations of security; and they exist because of vulnerabilities, that is, weaknesses in a system. There are two basic types of threats; accidental threats and intentional threats. An accidental threat can be manifested; and the results could either be an exposure of confidential information; or they could cause an illegal system state to occur - that is the modification of an object. Exposures can emerge from both hardware and software failures, as well as from the user; and any operational mistakes occurring thus result in the violation of confidentiality. It can also be manifested as the modification of an object, which is the violation of object integrity. An object in this regard can be both information and resource. An intentional threat is an action performed by an - entity with the intention to violate security. Examples of common attacks into a system are: interruption, interception and modification. The schematic of normal flow of information in a system without an attack is shown in Figure 2.4; while the three common attacks are presented in Figures 2.5 (a) to (c) adapted from Marti et.al. [27].
Figure 2.4: normal flow of information

Figure 2.5: (a) interruption of information in a system
Figure 2.5: (b) interception of information in a system

Figure 2.5: (c) modification of information of the system

Interruption, interception and / or the modification of information could occur as an attack in the flow of information into a system.
According to Williams and Fulp [28], in their journal article entitled: “A Biologically-Modelled Intrusion-Detection System for Mobile Networks”; the ant behavior was observed; and research was conducted in building an intrusion-detection system. Ants in particular are known to provide an attractive model for solving problems in the biological realm. Their behaviour while scavenging for food, in order to feed others in the hive and thereby preserving the population is both unique and robust. Furthermore, a population of ants functions as social creatures; that is, they are concerned with the survival of their entire population, rather than that of one member. Despite the simplicity of each ant in terms of brainpower, ant colonies develop into complex societies capable of rapidly adapting to environmental changes. An ant which happens to come across a source of food, will immediately return to the hive in a relatively direct fashion. As the ant returns, it leaves a pheromone on the terrain behind it. When other ants are scavenging and finding a trail of the pheromone; it is likely that they will begin to follow that pheromone to the previously discovered food source. Furthermore, ants returning from the hive to the source of food will also leave a pheromone trail, offering the potential for discovering more efficient paths to the food source. Over time, ants will follow the trails with the strongest pheromone, that is, the trails most-often traversed by other ants. As such, the majority of the scavenging ant population will be following an optimal path to a food source. This efficient behaviour is a key reason for the variety of ant populations that can be found thriving worldwide. When an agent takes the place of ants, and is given a simple task to perform, instead of scavenging for food, he wanders randomly and samples simple-system metrics at each node he traverses. Each agent is only capable of identifying one specific metric. Upon detecting a potential threat; the agents leave pheromones; they move away from the node in question. While focusing on the pheromone aspect of ant-colony optimization (ACO) the system moves towards the swarm’s intelligence realm; and the adaptability of ACO is more important to this security approach than merely exploring a problem space. This pheromone is able to attract other agents to the node, which might have an issue; and reinforces the pheromone’s strength leading- thereby leading agents to the node in question. In combination, these simple metrics form a behaviour-based intrusion-detection system capable of identifying any generalized threats to a system [28].

A research study on intelligent data-mining techniques for intrusion-detection models on a network was conducted by Mohammed and Arul-Lawrence [29], in which data-mining
techniques were employed for intrusion-detection purposes. Data-mining techniques have been successfully applied in almost all the fields, including process control, marketing, manufacturing, fraud detection and network management. A number of research projects have also applied data-mining techniques to various problems in intrusion detection [29].

In information security, intrusion detection is the act of detecting actions that attempt to compromise the three pillars of network security. These are confidentiality; data integrity; and resource availability. Intrusion detection does not, in general, include the prevention of intrusions. [29] present a new idea on how data mining can be applied to detect the intrusions. The various data-mining techniques can be classified, based on their functions, representation, preference criterion and algorithms. The main function of the model, in which one is interested in is the classification, as malicious or as a particular type of attack. Furthermore, it highlights the link and sequence analysis. Additionally, the data-mining system provides the means to easily perform data summarization and visualization, thereby assisting the security analyst; and also identifying the areas of concern. Common representations for data-mining techniques include rules, decision trees, linear and non-linear functions, including neural networks, instance-based examples and probability models [29].

Sindhu et.al. [30] additionally, in their journal article: “Decision-tree-based-light weight intrusion detection using a wrapper approach”: their method was based on constructing a lightweight intrusion-detection system for the detection of anomalies in the network. However, the most important aspect of building lightweight IDS depends on the pre-processing of the network data, identifying important features; and in the design of efficient-learning algorithm that classify normal and anomalous patterns. The design was looked into in three perspectives: removing redundant instances that cause the learning algorithm to be unbiased, identifying a suitable subset of features by employing a wrapper-based-feature selection algorithm; and by realizing the proposed IDS with a neural tree, in order to achieve better detection accuracy. The lightweight IDS has been developed by using a wrapper-based-feature-selection algorithm that maximizes the specificity and the sensitivity of the IDS, as well as by employing a neuron ensemble decision-tree iterative procedure to evolve the optimal features. The experimental set-up was based on six family decision-tree classifiers, these are: the Decision Stump, C4.5, Naive
Bayes’ Tree, Random Forest, Random Tree and the Representative Tree model to perform the
detection of the anomalies in the network pattern. The system has four phases, namely: Phase I
which is the pre-processing of the network traffic pattern; phase II, which is a feature extraction;
phase III which is the post-processing; and lastly, phase IV, which is the classification of the
traffic patterns. These were used in their entirety to detect any form of anomalous intrusion in the
network [30].

Furthermore, Zhijie and Ruchuang [31] in their article: “Intrusion Detection for Wireless Sensor
Network Based on Traffic-Prediction Model”, they based their method on exploring the Traffic
Prediction for the large-scale tiled WSN (Traffic Prediction Intrusion Detection Scheme, or
TPIDS), detecting attacks, which could have a considerable influence on the flow, such as the
selective forwarding attacks and DOS attacks. Their work focuses on two major sub-headings:
firstly, according to the flow characteristics of WSN, it designs a WSN traffic-prediction model,
by using Markov, which makes less computational complexity; and improves the forecast
accuracy; secondly, as the traffic prediction model used in the WSN Intrusion-Detection System,
it designs a threshold value beyond that of a judgment algorithm, based on the prediction to
detect any anomalies, followed by a self-similar characteristics analysis mechanism to judge the
reasons for the abnormal flow. If anomalies are caused by the attack, it would send the alarm
message to the source node. In their approach, there are two major advantages over other
approaches that have been used in intrusion detection. Firstly, TPIDS, which is based on the
anomaly-detection technology can detect unknown attacks; so it is more applicable to attack the
WSN, which has a diversity of variant and unpredictability. Secondly, TPIDS, adopts a fully
distributed scheme, so each node can independently complete traffic prediction and anomaly
detection and intrusion judgment, based on the analysis characteristics of self-similar network
traffic; while collecting data at the same time. The scheme does not require additional hardware
support or any co-operation between the nodes. Simulation was used to obtain the results, which
show that the improved Markov model has a high prediction accuracy. The calculation of each
node with very low overhead, and compared with the intrusion-detection systems based on
cooperation , it can be faster to detect the behaviour of the intrusions, while reducing the costs of
the communications and the energy.
According to Wang et.al. [32] in their article: “Energy-aware and self-adaptive anomaly detection scheme based on network tomography in mobile ad hoc networks”, they use an anomaly-detection method, which is used for securing services in mobile ad hoc network (MANET). They stated that a highly secure mechanisms results in high consumption of network resources, resulting in poor network performance. Therefore, in order to move the intrusion detection from its current state of security-centric design approach to network performance centric-design approach, they use an anomalous framework to design energy-aware and self-adaptive anomaly detection for resource constraints in MANETs. The system uses network tomography, which is a new technique for studying internal link performance. It is solely based on end-to-end measurements. With the support of a module comprising a novel spatial-time model to identify the MANET topology, an energy-aware algorithm to sponsor system service, a method based on the expectation maximum to infer delay distribution, and a self-organizing Map (SOM) neural network solution to profile link activity, the system is capable of detecting connection anomalies and localizing malicious nodes. Therefore, the scheme offers a balance between the overall network security and network performance, without overloading the network.

Shaban et.al. [33] in their article entitled: “Intrusion-Detection System in wireless ad-hoc Networks Based on Mobile-Agent Technology” address the problem in wireless ad-hoc networks, which are characterized by the lack of centralized infrastructure; as it is found in the wired network. They used a method based on mobile-agent technology to perform authentication mechanism; and hence, to improve the utilization of the network bandwidth; and to provide an efficient profile management in mobile ad-hoc networks. The mobile agent serves as the authentication mechanism in which, before any node can communicate with another, there must be an authentication, in order to determine whether it is really the authentic node that wants to communicate. The mobile agent uses two types of security techniques in the authentication processes. These are: a Destination-sequenced distance vector routing protocol (DSDV) and an Encryption model, which is based on asymmetric cryptography [33].

The rate at which intrusion occurs in networks is increasing at an alarming rate; and coupled with the unending features the internet present to us from our daily basis - with a very high frequency
traffic flow of the network, most of these IDSs designed cannot cope with such a frequency. Therefore, they cannot detect any intrusion; because the flow of the network traffic is too fast; and consequently, they decided to build a “two grains level intrusion detection system” (fine-grained and coarse-grained) to overcome this problem, the result of which has yielded detection rate of more than 93%, with an average processing speed that is equal to $3 \times 10^6$ seconds per example [34].

A very Fast Decision-Tree (VFDT) algorithm was used as a fast classifier. The purpose of the system is to process and analyze very high-speed network traffic, discovering and perfectly identifying any new attacks to decrease the false alarms to a minimum, and detecting the intrusion in real time. This was able to reduce the rate of false alarms; and also to solve perfectly the problem of very fast network traffic, which most IDSs designed have limitations. They were able to catch up with the limitations by detecting attacks as fast as the speed of the traffic flow. Asaf et.al., [34] in their work entitled “intrusion detection for mobile devices using a knowledge-based, temporal abstraction method”, they were able to develop a security ontology against the mobile malware and other malicious software that might want to run on the Smartphone by collecting raw data, such as time stamp, the number of sent SMSs and the events, such as software installation on the system. All of these were deployed into the knowledge-based, temporal abstraction (KBTA) method.

A light weight host-based IDS, combined with the central-management capabilities for Android-based mobile phones was designed. The IDS was installed on five different Android devices; and they were deliberately infected with malware; since no known malware has ever existed on Androids. The outcomes of the system developed provide an effective intrusion detection in detecting malicious applications running on the mobile devices with a detection rate of about 94%; and the CPU power consumption was just 3% of the average rendering the system good to use for an average Smartphone [35].

Since IDS is a security agent that is designed and deployed in both mobile and wired ecosystems to inspect and monitor network traffic against known attacks (signature-based approach) in the database; so that if a known one is detected, the alarm can trigger to alert the security experts. Notwithstanding all these approaches that have been used before now, there is still a problem in
the area of large-scale networks, which a number of the intrusion-detection systems have to be deployed because of the size of the network; and this is very expensive. Taejn et.al., [36], in their article entitled “Suspicious traffic sampling for intrusion detection in a software-defined network”, they proposed a traffic-sampling strategy for software-defined networking (SDN), whereby the SDN comprises the SDN controller and the OF- enabled switches on the control panel. They used a mirroring method by mirroring the data flow at each of the switches. This is fully configured by the SDN controller; and it is then sent to the IDS. The IDS then inspects all the data packets mirrored; and it generates an alarm to the experts - if any malicious path is found. Furthermore, this is done in such a way that the volume of the mirrored-sampled traffic flow at the switch sent to the IDS is less than the processing capacity of the IDS; because the IDS would not be able to inspect the mirrored-sampled traffic flow - if the volume sent is greater than the processing capacity of the system. The work was able to increase the inspection performance of malicious traffic in a large-scale network. The malware injection into a mobile phone is hereby presented in Figure 2.6.

3. Phone infected with Android virus from computer

2. Phone connected to computer using USB cable

1. Android virus infects computer

Figure 2.6: Malware injections in a mobile phone adapted from [36]

In addition, a novel behaviour-based anomaly-detection system for detecting significant deviations in mobile network activities was proposed in [37]. The main aim of the system is to
protect the mobile App users and their service provided from malicious applications. Through identifying malicious attacks or hidden applications that might have been installed on the device and specifically to detect a new mobile malware, which is self-updating; it was found on the certified Google Android marketplace, which the conventional intrusion detections, such as anomalies cannot detect. In their work; the detection was successful, based on the application of network traffic patterns to the system. In every application, a model that is used to represent a specific pattern was discovered locally (for example on the devices); and this, joined with a semi-supervised machine-language process, was used for learning the behavioural patterns to identify any deviations from the expected behaviour of the application.

2.12 UNSOLICITED SHORT-MESSAGING SERVICE (SMS)

Unsolicited SMSs are short-message services sent to users’ phones - without the user’s knowledge for the purpose of stealing their money, launching phishing attacks and causing damage to the phone by sending loads of SMSs to the phone. This can lead to the crashing of the phone. There have been many complaints about unsolicited premium SMSs advertisements globally, this is purposely done, in order to steal users’ money. While the unsolicited SMS is being received, the user’s money is being deducted automatically (even when the user has not subscribed for it). When user tries to opt out of the options, it becomes very difficult and impossible. In this light, an App was developed, which was installed on the phone. Whenever such an SMS wanted to come through to the mobile phone, the user would need to be prompt in asking for permission: either to allow or reject the SMS. If the user allows it, the message gets displayed in the inbox of the phone; otherwise it gets discarded; and the number is blocked permanently.

2.13 ELECTRONIC COMMUNICATIONS AND TRANSACTION (ECT) ACT

According to the ECT Act 2011, to date chapter V11 section contains the law, and the consequences for anyone who breaches it.

“Unsolicited goods, services or communications”

(1) Any person who sends unsolicited commercial communications to consumers, must provide the consumer:
(a) With the option to cancel his or her subscription to the mailing list of that person; and
(b) With the identifying particulars of the source from which that person obtained the consumer's personal information, on the request of the consumer.

(2) No agreement is concluded in which a consumer has failed to respond to an unsolicited communication.

(3) Any person who fails to comply with or contravenes the subsection;

(1) is guilty of an offence and liable, on conviction, to the penalties prescribed in section 89(1).
(4) Any person who sends unsolicited commercial communications to a person, who has advised the sender that such communications are unwelcome, is guilty of an offence; and is liable, on conviction, to the penalties prescribed in section 89(1).

(1) A person convicted of an offence referred to in sections 37(3), 45(1&2), 58(2), 80(5), 82(2) or 86(1), (2) or (3) is liable to a fine or imprisonment for a period not exceeding 12 months [38].

According to [39], there have been numerous complaints and arguments posted on the website about unsolicited SMSs from different companies. Presented here are some of the complaints from users.

a. “TIRED OF RECEIVING MESSAGING FROM 0820090000

“Please get rid of these messages from 0820090000, I am tired, I received more than 20 now, I don’t have time to keep on deleting this SMS, they always say messages cannot be deleted, get rid of them”

b. “VODACOM LIFE SPAM ME WITH SMS

“I’ve received a SMS from Vodacom Life earlier to receive an insurance quote. Number: 27 83920658055606. I do not care that you obtained my details from a database containing customer’s details. You contacted me from the following number: 27 11 558 4172. I am not interested in Vodacom Life or any other company phoning. DO NOT PHONE ME!”

c. “On the 2nd of August I sent Vodacom a request to stop receiving their illegal communications regarding the Vodacom millionaire’s competition to my 072........ Number. I explained why it is illegal (ECT Act 2004, section 45 (1) (a)) and that I
wanted to unsubscribe (chapter 45, section 4) Not only did I get reference a number (004Q-QS4Z); but I was told that they would contact me within 24 hours. I also requested that they put it in writing that they have in fact unsubscribed me. So far they have done nothing; I still receive the illegal SMS and they have not contacted me to date."

d. “This branch keeps sending spam unsolicited SMS about being pre-approved for a new contract. I have REPEATEDLY requested to be taken off all the marketing lists they have and believe this marketing communication breaches the Electronic Communication and Transaction Act. ” [39].

2.14 SUMMARY

In summary, this chapter has presented an overview of (IDS) Intrusion-Detection System: both in mobile phone and networked-based. Having critically reviewed the relevant literatures, it can be established that a lot has been done in the field of IDS, both in mobile devices and networked-based intrusions; but as far as the author knows, building an App to detect unsolicited SMS in mobile devices has not yet been done. This makes this study innovative, relevant; and could be applied in everyday life.
3.0 CHAPTER THREE

3.1 INTRODUCTION
This chapter gives insight into the App that has been developed; how it was developed; the versions targeted; and the codes that have been written. The important codes used in carrying out this work has been imported.

3.2 OVERVIEW OF UJSMS ANTIVIRUS
UJSMS Antivirus is an android application developed using Android studio with Java programming language. When the App is installed on the phone, it has the capacity to detect incoming unsolicited SMSs, any number that is not in the contact list of the mobile phone, it will block. Among the features of the App, there is an interface, where user can either block the SMS or allow it. If the user chooses to allow it; the SMS then goes into the inbox messages of the phone; otherwise, the SMS gets blocked. The versions targeted for the purpose of this study are version 4.4 – 6.0.1; because they cover a lot of the population up to about 1 billion [40]. The App was successfully installed on the various versions targeted and tested; and it worked as expected, as expected. Table 3.1 presents the history of android versions.

Table 3.1: History of android versions adapted from [40].

<table>
<thead>
<tr>
<th>Code Name</th>
<th>Version Name</th>
<th>Initial Release Date</th>
<th>API Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>No code name</td>
<td>1.0</td>
<td>September 23, 2008</td>
<td>1</td>
</tr>
<tr>
<td>Petit four</td>
<td>1.1</td>
<td>February 9, 2009</td>
<td>2</td>
</tr>
<tr>
<td>Cup cake</td>
<td>1.5</td>
<td>April 27, 2009</td>
<td>3</td>
</tr>
<tr>
<td>Donut</td>
<td>1.6</td>
<td>September 15, 2009</td>
<td>4</td>
</tr>
<tr>
<td>Éclair</td>
<td>2.0 – 2.1</td>
<td>October 26, 2009</td>
<td>5-7</td>
</tr>
<tr>
<td>Froyo</td>
<td>2.2 – 2.2.3</td>
<td>May 20, 2010</td>
<td>8</td>
</tr>
<tr>
<td>Ginger bread</td>
<td>2.3 – 2.3.7</td>
<td>December 6, 2010</td>
<td>9 - 10</td>
</tr>
</tbody>
</table>
3.3 APPLICATION INTERFACE

The UJSMS Antivirus application works in such a way that if there is an incoming unsolicited SMS from a phone number that is unknown (not in contact list of the phone), the application will automatically send the SMS to the spam; and it will notify the user that there is an SMS from an unknown contact. Among the features of the App, there is an interface for the user to either block or allow, it. If the user chooses to block, the SMS is then permanently deleted; but if the user chooses to allow it, the SMS will be displayed in the inbox of the user. This gives the user the opportunity to preview the first line of the message, to ensure whether she/he is interested or not. Secondly, if the contact is on the phone, the SMS will show in the inbox the message of the phone, thereby immediately prompting the user to read it.

3.4 ANDROID OPERATING SYSTEM

Android is a mobile operating system (OS) that was designed by Google, which was based on the Linux kernel. The design was specially designed for touch-screen mobile devices such as tablets and Smartphone. The user interface for Android is based on direct manipulations, using a touch screen, such as swiping, pinching and tapping coupled with a virtual keyboard to input the text. In further development, Google has also developed Android TV for televisions, Android
wearables and android auto for cars. Furthermore, notebooks, digital cameras and games can also make use of Android.

Android was originally developed by Android Inc., which was later bought over by Google in 2005. Android was then launched in 2007 with the initiation of Open Handset Alliance (OHA), a conglomerate of software, hardware and the telecommunication companies, which are eager to develop the open standards for mobile devices. The first commercial release was launched on the market in September 2008; and since then, there have been several releases ranging from 1.0 released in September 2008 to 7.0 released August 22, 2016. Android allows writing codes in Java language; and it has its own Dalvik Virtual Machine (DVM), which is used to execute the Android application.

Android applications can be downloaded from the Google play store, which consists of over 2.7 million apps as at February 2017. Currently, Android has the best selling operating system on tablets and smartphones - with over two billion active users. The successful story of Android has made it possible to have a goal for patent and copyright litigation between the technology companies. The different versions have been named after nice, sweet, and jelly names that renders them more presentable and preferred [41].

Android version 1.6 was released on September 15, 2009 with Application Program Interface (API) level 4, with each of the versions named after nice, yummy, and sweet things. This one was named “Donut”. It has the basic features that an Android phone would have, like camera, quick search box; and this provides information at the user’s finger tip - with just a click; and it also comes in different sizes and shapes etc, when released. Mobile-phone users love it; and they use it; but as newer and more recent versions were released; they tended to shift away from this; and to get the next version, which is Android 2.1; because there are some new and improved features present in the later version [42]. Figure 3.1 shows the pictorial diagram of the Android 1.6 version.
Android version 2.1 was released on October 26, 2009 with an API level of 5-7. Its sweetie name was “Éclair”; however, it also comes with its own unique features and functions with high density wallpapers, touch-screen function and drive-on navigation; but these features have been further improved over the version 1.6. Mobile-phone users used it; and they later shifted to the next available version, which is Android version 2.2. Figure 3.2 shows a pictorial diagram of the Android 2.1 version.
Android 2.2 was released on May 20, 2010 with API level of 8, named “Froyo”. It has been found to be fast on screen features like the other Android phones, with lightning-fast phone, and hotspot capabilities that make users connected everywhere they go, as well as lovely wallpapers. Mobile users also used it; but they later preferred the next version, which is 2.3. Figure 3.3 shows a pictorial diagram of the Android 2.2 version.
Android 2.3 was released on December 6, 2010 with API 9-10; and it is called the “Gingerbread”. The gingerbread comes with lots of improved functions over the previous versions; and such features include simple and faster to use for both the mobile phone users and the developer, together with long-lasting battery life and more improved gaming system. Obviously, mobile users tend to enjoy the experience they get, rather than the previous versions. But when android version 3.0 was launched, they moved to the latest version. Figure 3.4 shows a pictorial diagram of the Android 2.3 version.

![Android 2.3 Gingerbread](image)

Figure 3.4: Android 2.3 version adapted from [42]

Android version 3.0 was released on February 22, 2011 with API level 11-13; the name comes from honey, “Honeycomb” informing the mobile users that the packages in it are as sweet as honey. Its features considerably more improved on the previous versions. Honeycomb ushered in the era of tablets with a flexible interface design - with an imaginary big character to a faultless user experience. Figure 3.5 shows a pictorial diagram of the Android 3.0 version.
Android 4.0 was launched on October 18, 2011 with API level 14-15. It was called “Ice Cream Sandwich”. The Ice cream sandwich stepped up on customization and user control. Users can now manage and organize their App, as they wish, knowing how much data users have used; and they can share information and the data wherever they wish to. Figure 3.6 shows a pictorial diagram of the Android 4.0 version.

Figure 3.5: Android 3.0 version adapted from [42]

Figure 3.6: Android 4.0 version adapted from [42]
The Android 4.1 version was launched on July 9, 2012 with API 16-18 it was given the name of “Jelly Bean”. Jelly bean has considerably improved functions over the Android version 4.0. Android 4.1 comes with more intelligent packages; and users can now personalize their mobile device, notification more actively; and it allows one device to work for multiple accounts etc. Figure 3.7 shows a pictorial diagram of the Android 4.1 version.

Figure 3.7: Android 4.1 version adapted from [42]

The Android version 4.4 was released on October 31, 2013 with API level 19-20. The name is Kitkat; and with Kitkat, the mobile-phone users can get things done faster especially with internet and Google search. Users can now have a voice search by just sending a text, getting directions, playing songs. Furthermore, it comes in more fashionable designs. Figure 3.8 shows a pictorial diagram of the Android 4.4 version.
Android 5.0 was released on November 12, 2014 with API level 20-22, its juicy name is “Lollipop”; and with lollipop there are Android versions for television, the wearables, tablets, phones and cars. It also has a bold visual approach and a juicy physical response of material design etc. Figure 3.9 shows the pictorial diagram of the Android 5.0 version.

Android 6.0 was released October 5, 2015 with API level 23; it is called “Marshmallow”. This has a big advantage over the previous versions due to its uniqueness and a more improved system design; and it is user friendly. Some of the features include easy shortcuts with just a tap, easily swiped screen, a long-lasting battery; and a new App permission for more control for mobile phone users etc. Figure 3.10 shows the pictorial diagram of the Android 6.0 version.
Android 7.0 was launched on August 22, 2016 with API level 24-25; and it is called “Nougat”; and it is the latest Android version that has been released so far. It is very easy to use. This comes with many upgraded functionalities, like speak-search, fast shortcuts, fast touch screen, lots of App that does virtually anything you want ranging from school work to businesses, office tools, fashions and styles, advertisements etc. Figure 3.11 shows pictorial a diagram of the Android 7.0 version
3.5 ANDROID ARCHITECTURE

The Android architecture comprises four layers. Each layer has its own unique function and role to play in the architecture. The Linux kernel is the first layer, which serves as the base for all the other layers; this consists of Apps like the power management, camera driver, keypad driver etc. The second layer consists of the libraries and the Android run-time; the libraries consists of Apps, such as the surface manager, the media framework, webkit etc; while the Android run-time comprises the core libraries and the Dalvik Virtual Machine (DVM). The third layer, which is the application layer consists of the activity manager, the package manager, the telephony manager, the notification manager etc; while the last layer, which is the applications, serves as the interface between the mobile-phone user and the applications, such as the home, contacts, browser and phone [43].
3.5.1 LINUX KERNEL LAYER

The Linux kernel layer serves as the basis for all the other applications in the Android architecture. This Android architecture is mainly based on the Linux kernel; and it is the core part of the Android architecture, which includes services like: the memory management and the power management etc. This layer also provides hardware and software binding for better and more effective communication [43].
3.5.2 LIBRARIES

Android has its own specially designed library for the applications, just as we have other libraries like the web libraries to access the web browsers etc. Android libraries are written in C/C++, which cannot be accessed directly by other applications - except with the help of the application framework.

3.5.3 ANDROID RUN-TIME

Android run-time was designed largely for Android’s mobile devices and tablets to be able to run in an embedded real-world environment; where there are limited memory, limited batteries etc. Dalvic is used in executing the Androids’ applications on android devices, which makes it an integral part of Android. The programs are normally written using Java programming language and compiled, according to various codes. The core library is also written in Java programming.
language, which comprises all the classes, utilities and tools one is expected to use when working with Android.

### 3.5.4 ANDROID RUN-TIME

![Android Runtime](image)

Figure 3.15: Android Runtime adapted from [43]

### 3.5.5 APPLICATION FRAMEWORK

The application framework is the toolkit that all the other applications use; it is written in Java programming language. This includes applications that come from Google or the ones that come with the phone, like the home App, or can be the ones that are written. The entire App uses the same structure and same APIs (Application Programming Interface).
A. **Activity manager**: The Activity manager manages the overall activities of the App, such as the lifecycle; and she/he maintains proper management of the application.

B. **Resources manager**: The Resource manager provides access to the general layout of the App, such as customary or standard layout and the graphics of the App.

C. **Notification manager**: The Notification manager allows applications to display conventional alerts in the status bar.

D. **Location manager**: The Location manager promptly displays the alert when the users change locations, or enter a specific geographical location.

E. **Package manager**: The Package manager is used to retrieve the information about application packages currently installed on the device [44].

F. **Window manager**: The Window manager takes care of displays, views and layouts of the App.

G. **Telephony manager**: The Telephony manager is basically used to handle anything relating to the settings of network connections on the device.

### 3.5.6 APPLICATIONS

The Application layer is the final and the upper most layer of the Android architecture, which includes the home App, the contact App, the phone App and the browser App. Applications, like camera, google map, SMSs, MMs, contacts, the calendar etc are able to work with the end-users through the applications.
3.6 ANDROID’S BASIC-APPLICATION COMPONENTS

Application components are the important entry points of an Android application. Each unit of the components represents different entry points, which the system can use to get into the application. Although not all the components are used as entry points into the application; rather they depend on each other; and they can also stand as single entities to perform a particular task. Therefore, each component is an important building block that helps to describe the overall performance of the application. There are four major types of application components; each component performs a unique function and has a unique lifetime that defines how the component is formed; and also how it is being destroyed [45].

The four major types of application components are hereby presented:

3.6.1 Activities: An activity is a single screen with a user interface, a screen that users can work with at a pace that allows one to perform a single task. For example, in the WhatsApp application, we have one activity, like chatting, another activity like videos calling one another, like voice recording; and another one like camera; all these functions work together to give a coherent WhatsApp application. An activity is implemented as subclass of Activity.
3.6.2 Service: A service is a component that runs at the background of an application to perform work. A service is a background component that allows you to work on an activity. It does not have a user interface. For example, a service can be playing music in the background; while a user still works on another activity. A service does not disturb an activity from working. Hence, it is a background long-time running component.
3.6.3 Content providers: A Content provider is a component that manages and oversees a preferred set of application data; and it makes it available for other application to use; it can be stored on the web or in a file system; where the application can easily access to it. However, other applications can use, update, modify and query the content provider (provided the application has appropriate the permission). For example, the Android system uses a content provider that manages the user’s contact information. So, any application with the content provider’s permission can query any part of the content provider to read and write information about a specific person, such as: (ContactsContract.Data). A content provider is also useful in storing confidential data that are private to the application.
3.6.4 Broadcast Receiver: A *broadcast receiver* is a component that receives and reacts to a broadcast announcement; while the broadcast originates from the system. For example, there can be a broadcast of a system service, such as a low battery, or a screen that is locked. An application can also send a broadcast to others to listen to and use telling them there are new data downloaded to the device. As such, a broadcast receiver does not provide a user interface; but instead, it creates a notification bar to notify the user whenever a broadcast is received. The application is given 10 seconds to fix a time to react to the broadcast.
Figure 3.21: Broadcast receivers Adapted from [45]

Figure 3.22: Android broadcast receiver
3.7 ARCHITECTURAL LAYOUT OF THE GLOBAL SYSTEM MOBILE (GSM) IN RELATION TO THE APP

The architectural layout of the GSM system in relation to the App developed is an important aspect we need to look into, in order to really understand the App; and how it works. The Base Transceiver System (BTS) is a part of the network equipment that facilitates the wireless communication between a device and the network. The BTS system transmits wireless networks to the phone, and from the phone to the Operating System OS of the Android, and then down to our UJSMS Antivirus. Then we have the broadcast receiver, where the intent is received and transmitted. The message receiver receives and stores it in the storage for future use [46]. The representation is hereby presented in Figure 3.23

![Diagram of architectural layout of the GSM system]

Figure 3.23: Architectural Layout of the GSM system adapted from [47]
3.8 APP FLOW CHART

When the SMS is received, it must be checked, to see whether it is in the phone contact list. If it is in the contact list of the phone, it will be inserted into the inbox; and the user will be notified. Then if it is not in contact of the phone, check the status of the contact. If allowed, it goes into the inbox. If it is blocked, it goes to end. Also, if there is a new spam it stores to new spam table and notify then it goes to end of the flowchart.

![Flowchart Image]

Figure 3.24: UJSMS Antivirus flowcharts
// set intent so it does not start a new activity
Notification note = builder.setContentIntent(contentIntent)
    .setSmallIcon(R.mipmap.ic_launcher)
    .setWhen(System.currentTimeMillis())
    .setContentTitle("New SMS Spam")
    .setContentText("A new SMS spam has been detected!\n" +
messageBody).build();
    note.flags |= note.FLAG_AUTO_CANCEL;
android.app.NotificationManagermNM = (android.app.NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
mNM.notify(notifId, note);
    } else if(Contact.Status(mAddress).equals("allowed")) {
insertMessageAndNotify();
    } else if(Contact.Status(mAddress).equals("blocked")) {
    //Dont do anything
Log.v("MsgReceived","This is a spam that has been blocked! don't do anything");
    }
}
    else{
insertMessageAndNotify();
    }
    }
}
The code for the blocked character of the algorithm is hereby presented:

```java
// Check if the phone number that has sent the sms is in the phone contact list
if (!Contact.contactnumberlist(context).contains(mAddress)) {
    // Check the status of the contact number is it allowed, it is blocked or is New spam
    if (Contact.Status(mAddress).equals("newspam")) {
        Storage.write(Storage.NEW_SMS, objSms);
        PendingIntentcontentIntent = PendingIntent.getActivity(context, 0,
        new Intent(context, SpamActivity.class), 0);
        NotificationCompat.Builder builder = new NotificationCompat.Builder(context);
```

Figure 3.25: UJSMS antivirus blocked character algorithm
3.10 UJSMS ANTIVIRUS IMPORTANT CODES

The codes that were written at the android studio are hereby imported into the work for clarity purposes.

3.10.1 BROADCAST RECEIVER FOR SMS

The broadcast receiver codes for the SMS are hereby presented:

<!-- Legacy broadcast receiver for SMS -->
<receiver
    android:name=".receiver.SmsReceiverLegacy"
    android:enabled="@bool/preKitKat"
    android:permission="android.permission.BROADCAST_SMS">
    <intent-filter android:priority="2147483647">
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>

3.10.2 DELIVERY RECEIPT

The codes for the delivery report are hereby presented:

<!-- Broadcast receiver for SMS -->
<receiver
    android:name=".receiver.SmsReceiver"
    android:enabled="@bool/hasKitKat"
    android:permission="android.permission.BROADCAST_SMS">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_DELIVER" />
    </intent-filter>
</receiver>

3.10.3 MAIN RECEIVER PARENT

This is the parent and the main receiver of the public class code:

public class SmsReceiverLegacy extends MessagingReceiver { 
}
3.10.4 MESSAGING RECEIVER MAIN RECEIVER

This is the part of the imported codes from “com.moez.QKSMS” it has features like App notifications, App power management, App content shared preferences, telephony. SMS Messages etc. all these are part of the UJSMS Antivirus codes.

```java
package com.moez.QKSMS.receiver;

import android.app.Notification;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.PowerManager;
import android.preference.PreferenceManager;
import android.support.v7.app.NotificationCompat;
import android.telephony.SmsMessage;
import android.util.Log;
import com.antivirus.Contact;
import com.antivirus.Sms;
import com.antivirus.SpamActivity;
import com.antivirus.Storage;
import com.moez.QKSMS.R;
import com.moez.QKSMS.common.BlockedConversationHelper;
import com.moez.QKSMS.common.ConversationPrefsHelper;
import com.moez.QKSMS.common.utils.PackageUtils;
import com.moez.QKSMS.data.Message;
import com.moez.QKSMS.service.NotificationService;
import com.moez.QKSMS.transaction.NotificationManager;
import com.moez.QKSMS.transaction.SmsHelper;
import com.moez.QKSMS.ui.settings.SettingsFragment;
import org.mistergroup.muzutozvednout.ShouldIAnswerBinder;
import java.util.Date;

public class MessagingReceiver extends BroadcastReceiver {
    private final String TAG = "MessagingReceiver";

    private Context mContext;
    private SharedPreferences mPrefs;
```
private String mAddress;
private String mBody;
private long mDate;

private Uri mUri;

@Override
On receiving the intent function, this is the function that gets called when an SMS is received.

public void onReceive (Context context, Intent intent) {
    Log.i(TAG, "onReceive");
    abortBroadcast();

    mContext = context;

    The function here initializes the shared preferences.
    mPrefs = PreferenceManager.getDefaultSharedPreferences(context);

    This convert PDU Received to SMSMessage Object.
    if (intent.getExtras() != null) {
        Object[] pdus = (Object[]) intent.getExtras().get("pdus");
        SmsMessage[] messages = new SmsMessage[pdus.length];
        for (int i = 0; i < messages.length; i++) {
            messages[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
        }
        SmsMessagesms = messages[0];
        if (messages.length == 1 || sms.isReplace()) {
            mBody = sms.getMessageBody();
        } else {
            StringBuilder bodyText = new StringBuilder();
            for (SmsMessage message : messages) {
                bodyText.append(message.getMessageBody());
            }
            mBody = bodyText.toString();
        }
    }

    Get Phone number from the SMS object
    mAddress = sms.getMessageOriginatingAddress();

    Get Date from the SMS object
    mDate = sms.getTimestampMillis();
if (mPrefs.getBoolean(SettingsFragment.SHOULD_I_ANSWER, false) && PackageUtils.isAppInstalled(mContext, "org.mistergroup.muzutozvednout")) {

ShouldIAnswerBinders shouldIAnswerBinder = new ShouldIAnswerBinder;
shouldIAnswerBinder.setCallback(new ShouldIAnswerBinder.Callback() {
    @Override
    public void onNumberRating(String number, int rating) {
        Log.i(TAG, "onNumberRating " + number + ": " + String.valueOf(rating));
        shouldIAnswerBinder.unbind(context.getApplicationContext());
        if (rating != ShouldIAnswerBinder.RATING_NEGATIVE) {
            insertMessageAndNotify();
        }
    }

    @Override
    public void onServiceConnected() {
        try {
            shouldIAnswerBinder.getNumberRating(mAddress);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public void onServiceDisconnected() {
    }
});
shouldIAnswerBinder.bind(context.getApplicationContext());

} else {

    //Main System Logic
    //Get the phone number from the sms received ***************
    final String messageBody = sms.getMessageBody();
    String cellnumber = mAddress;
    int notifId = 1234;
    SmsobjSms = new Sms();
    objSms.setAddress(mAddress);
    objSms.setMsg(mBody);
    objSms.setTime(new Date(mDate).toString());
    objSms.setLongTime(mDate);
    objSms.setId(Long.toString(mDate));

    //end get the phone number and other parameters from the sms received
// Check if the phone number that has sent the sms is in the phone contact list
if (!Contact.contactnumberlist(context).contains(mAddress)) {
// Check the status of the contact number is it allowed, it is blocked or is New spam
if (Contact.Status(mAddress).equals("newspam") ) {
Storage.write(Storage.NEW_SMS, objSms);
PendingIntentcontentIntent = PendingIntent.getActivity(context, 0,
new Intent(context, SpamActivity.class), 0);

NotificationCompat.Builder builder = new NotificationCompat.Builder(context);

// set intent so it does not start a new activity
Notification note = builder.setContentIntent(contentIntent)
 .setSmallIcon(R.mipmap.ic_launcher)
 .setWhen(System.currentTimeMillis())
 .setContentTitle("New SMS Spam")
 .setContentText("A new SMS spam has been detected!
" + messageBody).build();

note.flags|= note.FLAG_AUTO_CANCEL;
android.app.NotificationManagermNM = (android.app.NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
mNM.notify(notifId, note);
} else if (Contact.Status(mAddress).equals("allowed") ) {
insertMessageAndNotify();
} else if (Contact.Status(mAddress).equals("blocked") ) {
// Don't do anything
Log.v("MsgReceived", "This is a spam that has been blocked! don't do anything");
}
} else{
insertMessageAndNotify();
}

The functions here display Notification to the user

private void insertMessageAndNotify() {

mUri= SmsHelper.addMessageToInbox(mContext, mAddress, mBody, mDate);

Message message = new Message(mContext, mUri);
ConversationPrefsHelperConversationPrefs = new ConversationPrefsHelper(mContext,
message.getThreadId());

if (BlockedConversationHelper.isFutureBlocked(mPrefs, mAddress)) {

BlockedConversationHelper.unblockFutureConversation(mPrefs, mAddress);
BlockedConversationHelper.blockConversation(mPrefs, message.getThreadId());
message.markSeen();

BlockedConversationHelper.FutureBlockedConversationObservable.getInstance().futureBlockedConversationReceived();

// If we have notifications enabled and this conversation isn't blocked
} else if (conversationPrefs.getNotificationsEnabled() && !BlockedConversationHelper.getBlockedConversationIds(

PreferenceManager.getDefaultSharedPreferences(mContext).contains(message.getThreadId()))
{
       Intent messageHandlerIntent = new Intent(mContext, NotificationService.class);
messageHandlerIntent.putExtra(NotificationService.EXTRA_POPUP, true);
messageHandlerIntent.putExtra(NotificationService.EXTRA_URI, mUri.toString());
mContext.startService(messageHandlerIntent);

UnreadBadgeService.update(mContext);
NotificationManager.create(mContext);

} else { // We shouldn't show a notification for this message
message.markSeen();
}

if (conversationPrefs.getWakePhoneEnabled()) {
PowerManager pm = (PowerManager)mContext.getSystemService(Context.POWER_SERVICE);
PowerManager.WakeLock wakeLock = pm.newWakeLock((PowerManager.SCREEN_DIM_WAKE_LOCK | PowerManager.ACQUIRE_CAUSES_WAKEUP), "MessagingReceiver");
wakeLock.acquire();
wakeLock.release();
}

3.10.5 GET THE LIST OF ALL THE CONTACT NUMBERS

This function is used to get the list of all the contact numbers from the phone book, in order to achieve this contentresolver that queries the list of contact using contactscontract.contacts._ID as a key was used.

public static List<String> contactnumberlist(Context context) {
    List<String> numberlist = new ArrayList<String>();
ContentResolver cr = context.getSystemService(Context.CONTENT_RESOLVER);
Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

if (cur.getCount() > 0) {
    while (cur.moveToNext()) {
        String id = cur.getString(cur.getColumnIndex(ContactsContract.Contacts._ID));
        String name = cur.getString(cur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));

        if (cur.getInt(cur.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER)) > 0) {
            Cursor pCur = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null, ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " + id, null, null);
            while (pCur.moveToNext()) {
                String phoneNo = pCur.getString(pCur.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
                if (phoneNo != null)
                    numberlist.add(phoneNo);
            }
            pCur.close();
        }
    }
}
return numberlist;

3.10.6 READ AND WRITE THE MOBILE PHONE MEMORY

This is a simple class that was used to read and write to the phone memory.

package com.antivirus;

import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.util.Log;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

85
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

When one writes to the memory, it must have a unique identifier for the area of the memory you are using. In this case, the CONTACTS, NEW_SMS, DELETED_SMS; this class is just a wrapper for normal share preferences used in Android. The SMS object is stored as a json string in the shared preference, every time before reading we parse the json string back to a json array, in order to easily add or remove a value stored at a certain key.

public class Storage
{
    public static final java.lang.String CONTACT = "CONTACTS";
    private static SharedPreferences mSharedPref;
    public static final String NEW_SMS = "NEW_SMS";
    public static final String DELETED_SMS = "DELETED_SMS";
    public static final String BLOCKED_CONTACT = "BLOCKED_CONTACT";
    public static final String ALLOWED_CONTACT = "ALLOWED_CONTACT";

    public static void init(Context context)
    {
        if(mSharedPref == null)
            //mSharedPref = context.getSharedPreferences(context.getPackageName(),
            //Activity.MODE_PRIVATE);
            mSharedPref = PreferenceManager.getDefaultSharedPreferences(context);
    }

    public static String read(String key, String defValue) {
        return mSharedPref.getString(key, defValue);
    }

    public static void write(String key, String value) {
        SharedPreferences.Editor prefsEditor = mSharedPref.edit();
        prefsEditor.putString(key, value);
        prefsEditor.commit();
    }

    public static boolean read(String key, boolean defValue) {
        return mSharedPref.getBoolean(key, defValue);
    }
}
public static void write(String key, boolean value) {
    SharedPreferences.Editor prefsEditor = mSharedPref.edit();
    prefsEditor.putBoolean(key, value);
    prefsEditor.commit();
}

public static Integer read(String key, int defValue) {
    return mSharedPref.getInt(key, defValue);
}

public static void write(String key, Integer value) {
    SharedPreferences.Editor prefsEditor = mSharedPref.edit();
    prefsEditor.putInt(key, value).commit();
}

The SMS object is stored as a json string in the shared preference, every time before reading we parse the json string back to a json array in order to easily add or remove a value stored at a certain key.

public static boolean write(String key, Sms objSms) {
    //Retrieve the values
    String newSMSList = mSharedPref.getString(key, "[]");
    try {
        JSONArray jarray = new JSONArray(newSMSList);
        jarray.put(new JSONObject(objSms.JsonSMS()));
        SharedPreferences.Editor prefsEditor = mSharedPref.edit();
        prefsEditor.putString(key, jarray.toString());
        prefsEditor.apply();
        return true;
    } catch (JSONException e) {
        e.printStackTrace();
        Log.v("error", e.getMessage());
    }
    return false;
}
public static List<Sms> read(String key) {
    //Retrieve the values
    List<Sms>smsLst = new ArrayList<Sms>();
    String newSMSList = mSharedPref.getString(key, "[]");
    try {
        JSONArray jarray = new JSONArray(newSMSList);
        for (int i = 0; i < jarray.length(); i++) {
            JSONObject jsonObj = jarray.getJSONObject(i);
            try {
                if (!jsonObj.getString("sender").equals("")) {
                    smsLst.add(Sms.JsonToSMS(jsonObj.toString()));
                }
            }
            catch (JSONException e) {
                //Object is empty
                e.printStackTrace();
            }
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
    return smsLst;
}

public static void writeBlockedContact(String key, String address, String status) {
    //Retrieve the values
    String newContactList = mSharedPref.getString(key, "[]");
    String currentDateTimeString = DateFormat.getDateTimeInstance().format(new Date());
    try {
        JSONArray jarray = new JSONArray(newContactList);
        JSONObject obj = new JSONObject();
        boolean exist = false;
        for (int i = 0; i < jarray.length(); i++) {
            JSONObject jsonObj = jarray.getJSONObject(i);
            if (jsonObj.getString("sender").equals(address)) {
                jarray.remove(i);
                break;
            }
        }
        obj.put("status", status);
        jarray.put(obj.toString());
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
jobj.put("sender", address);
jobj.put("time", currentDateTimeString);
jobj.put("longtime", new Date().getTime());
jarray.put(jobj);

SharedPreferences.Editor prefsEditor = mSharedPref.edit();
prefsEditor.putString(key, jarray.toString()).commit();
}

public static List<Contact> readBlockedContact(String key) {
    //Retrieve the values
    List<Contact> contactLst = new ArrayList<Contact>();
    String newContactList = mSharedPref.getString(key, "["]");
    try {
        JSONArray jarray = new JSONArray(newContactList);
        for (int i=0; i<jarray.length(); i++) {
            JSONObject jsonObj = jarray.getJSONObject(i);
            Contact tempContact = Contact.JsonToContact(jsonObj.toString());
            if (tempContact != null && tempContact.getAddress() != null) {
                contactLst.add(tempContact);
            }
        }
    }
    catch (JSONException e) {
        e.printStackTrace();
    }
    return contactLst;
}

There is a list of blocked contacts in the memory; now we check whether the incoming SMS’s sender is blocked. We simply loop through the list of blocked messages; if the phone number is in the list, we return to new spam.

public static String CheckContact(String contact) {
    //Check if contact is blocked
    String key = Storage.BLOCKED_CONTACT;
    //Retrieve the values

List<Contact> contactLst = new ArrayList<Contact>();
String newContactList = mSharedPref.getString(key, "[]");
try {
JSONArray array = new JSONArray(newContactList);
for (int i = 0; i < array.length(); i++) {
    JSONObject jsonObj = array.getJSONObject(i);
    Contact tempContact = Contact.JsonToContact(jsonObj.toString());
    if (tempContact != null && tempContact.getAddress().equals(contact)) {
        return tempContact.getStatus();
    }
}
return "newspam";
} catch (JSONException e) {
    e.printStackTrace();
}
return "newspam";

The functions here are used in deleting an SMS from the memory.

public static boolean delete(String key, Sms objSms) {
    //Retrieve the values
    List<Sms> smsLst = new ArrayList<Sms>();
    String newSMSList = mSharedPref.getString(key, "[]");
    try {
        JSONArray array = new JSONArray(newSMSList);
        for (int i = 0; i < array.length(); i++) {
            JSONObject jsonObj = array.getJSONObject(i);
            Sms tempSMS = Sms.JsonToSMS(jsonObj.toString());
            if (tempSMS.getAddress() != null && tempSMS.getAddress().equals(objSms.getAddress())
                    && tempSMS.getMsg().equals(objSms.getMsg())) {
                array.remove(i);
                SharedPreferences.Editor prefsEditor = mSharedPref.edit();
                prefsEditor.putString(key, array.toString()).commit();
                return true;
            }
        }
    }
    } catch (JSONException e) {
    e.printStackTrace();
}
return false;
3.11 SUMMARY

In conclusion, looking through all the functions; they are almost the same; and it is either reading or writing to the shared preference/storage. Since shared preference cannot store objects; we first convert the object to a json string; and then, we store it in the memory. In order to read the object back we convert the json string back to a json array; and we then are able to retrieve the values. This chapter basically gives the codes written for the App; and it has been imported to the report. The code is a critical part of this study, which the write-ups, figures and experimental set-ups complements, in order to have a good, robust, comprehensive and workable application.
4.0 CHAPTER FOUR

IMPLEMENTATION

4.1 INTRODUCTION

In this chapter, the experiment set up and testing of the App on various versions of Android mobile phones is presented. The experiments are to verify the functionality and practicality of the App.

4.2 EXPERIMENTAL SET-UP

The application was installed on different versions of Android phones targeted in the coding. (4.4 – 6.0.1). For each version, the application was successfully installed on the phones by using the USB cable to transfer the App to the phones. An unknown number (number not in the phone contact list) was used to send SMSs to the phone. The application recognises the SMSs; and it shows on the notification bar informing the user that there is an incoming SMS. The user then clicks on the message. There is then a prompt message asking if the user wants to allow, or block the SMS. If yes, it goes into the messaging inbox otherwise it is blocked.

4.3 HOW TO LOAD THE APP INTO THE MOBILE PHONE

The Android phone is connected to the computer system by using a USB cable to load the application on the phone. Below are the necessary steps to be taken to install the application in the phone:

4.3.1 Step 1: CONNECTION

This section shows how to transfer the application from the laptop to the mobile phone before it is tested. Figure 4.1 shows the mobile-phone connection to the laptop through the USB connection.
4.3.2 Step 2: DEVICE SUCCESSFULLY INSTALLED

After connecting the USB cable; the device driver was successfully installed on the system, which displayed as device successfully installed. Otherwise one needs to go online to download Original Equipment Manufacturer (OEM) USB drivers. This will display all the various links to each of the companies, such as Techno, Samsung etc. Click on the brand of your phone and download. While downloading, check box “include folder” to make the driver complete so that the device can be connected to the computer, as shown in Figure 4.2

Figure 4.1: USB Connection to the laptop

Figure 4.2: Device successfully installed
4.3.3 Step 3: DEVICE READY TO USE

Go to settings on your phone; navigate to “developer option” if the phone is the Android version 4.4 to 6.0; but from 6.1 to 7.0, it is hidden under “about phone”. One needs to tap it seven times; then the developer option will be enabled. Then click on the option USB debugging, in order to be activated (debug mode when USB is connected), as shown here in Figure 4.3.

Figure 4.3: Enabling USB debugging
4.3.4 Step 4: ENABLE USB DEBUGGING

There will be a prompt message displaying on the phone - asking if you want to allow debugging; click on OK, as shown here in Figure 4.4

Figure 4.4: allowing USB debugging on the laptop

4.3.5 STEP 5: LAUNCH ANDROID STUDIO

Android studio is the interface in which the App is developed; and it runs into the mobile phone. After being successfully installed on the phone; the Android studio, which is already installed on the laptop, will be opened and ready to run on the mobile phone. Figure 4.5 presents the launching of the Android studio.
4.3.6 STEP 6: CLICK THE RUN BUTTON

Click on the **RUN** button on the Android-studio menu. Then the application starts running; and install it on the phone, as shown in Figure 4.6.
4.3.7 Step 7: GRADLE BUILD RUNNING

At this stage, the Gradle Build will start running and loading the application to the phone, as seen in the picture: Figure 4.7 with an arrow.

![Gradle building running](image)

Figure 4.7: Gradle building running

4.3.8 Step 8: APPLICATION SUCCESSFULLY INSTALLED

After the gradle build has successfully run and the application has been installed on the phone; the next phase that is going to show on the screen of the mobile phone will be application successfully installed on the phone. The name of the App will show on the mobile-phone screen “UJSMS”, as seen here in Figure 4.8.
4.4 TESTING THE APP ON THE ANDROID- PHONE VERSION 4.4

The application was tested on android phone version 4.4; and it works as expected. The normal procedures from 4.2.1 to 4.2.8 were followed, (how to load the application to the phone). The phone was connected to the laptop by using the usb cable; the phone driver was successfully installed on the laptop. After navigating to the phone settings, go to developer options to enable the USB debugging. In the next step, the Android studio was launched on the laptop; and the run button was clicked on the Android studio. It then displayed a window to show connected phones and to choose which phone to connect with in order transfer the application to it. The right phone was clicked, which is Samsung galaxy; and the application was loaded to the phone successfully, taking about 15-20 minutes, as seen in Figure 4.9.

Figure 4.8: App successfully installed on the phone
4.4.1 MAKING THE APPLICATION TO BE THE DEFAULT SMS

After successful installation, it gives room for settings, such as changing the color fonts and to make the UJSMS (Antivirus) the default messaging App. The App was made the default messaging App; the color was changed from green to orange, as presented in Figure 4.10.

4.4.2 VIEWING THE SETTINGS AND THE SPAM
This stage the App has been successfully installed; the UJSMS App has been made the default messaging App. The user can then view the settings, view spam etc. Figure 4.11 presents the viewing of the settings and the spam settings.

4.4.3 VIEWING THE SPAM INBOX

The user has the opportunity to view the spam inbox. This is where all the blocked SMSs will be stored - after the user has blocked them. Any number that has been blocked cannot send SMSs to the mobile phone any longer. Figure 4.12 presents the viewing of the spam inbox.
4.4.4 SMS SHOWS IN THE NOTIFICATION BAR

After successfully loading the application to the phone, a phone number, which is not in the contact list of the mobile phone was used to send a SMS to the phone. The SMS came in quite alright and shows at the notification bar, notifying the user that there is an incoming SMS, as presented Figure 4.13
4.4.5 ALLOW-OR-BLOCK FEATURE OF THE APPLICATION

After the SMS has been notified in the notification bar, indicating that there is a new incoming SMS, the user must click on the message; then it displays the options to: block or allow? If allowed, it goes immediately to the mobile-phone inbox messaging; otherwise it gets blocked, as presented in Figure 4.1.

![Image](image.png)

Figure 4.14: Allow or block features of the App

4.4.6 SMS DISPLAYED IN THE INBOX AFTER CHOOSING THE ALLOW OPTION

After the user has selected the allow option, the SMS goes to the mobile-phone inbox- messaging automatically, without any interference; and subsequently, it sends its SMS from this particular number, which will henceforth be allowed into the mobile phone inbox, messaging system. Figure 4.15 shows the picture of the demonstration performed in which the allow option was chosen.
4.4.7 CHOOSING THE BLOCK OPTION

For this section, according to the demonstration performed, the blocked option was chosen. When the SMS comes in at the notification bar, notifying the mobile phone user that there is an incoming SMS; the user then taps on the message: the App displayed block or allow, as seen in Figure 4.14. Then the blocked option was chosen after which, the message went to spam; and the number was blocked permanently. Figure 4.16 presents the picture.
4.4.8 TESTING THE APPLICATION WITH KNOWN CONTACT

The application was also tested with another phone number in the contact list of the mobile phone; and the SMS entered to the inbox messaging directly - without any interference or blockage; as seen in Figure 4.17: “Roseline TEST”.

![Image](image_url)

Figure 4.17: Testing with known contact

4.5 TESTING THE APPLICATION ON VERSION 5.0

The application was also tested on Android version 5.0; and it works as expected. The normal procedures for transferring the app to the device were followed, (as seen in 4.2.1 – 4.2.8); and it was successfully installed. An SMS was then sent to the phone, using an unknown contact, the SMS came in and was notified in the notification bar. The SMS was tapped; and the App displayed options of “allowed” or “blocked”; and the ‘allowed’ option was chosen; and the SMS displayed automatically in the inbox. Here are some screen shots to buttress our findings.
4.5.1 APP SUCCESSFULLY INSTALLED ON THE PHONE

The application was successfully installed on the mobile phone following the steps from 4.2.1-4.2.8, this time testing the App by using Android version 5.0. Figure 4.18 shows it successfully installed; as may be seen on the screen of the mobile phone.

![App successfully installed](image)

Figure 4.18: App successfully installed.

4.5.2 OPTION CHANGE THEME

There is an option to change the theme to any color, which the user wishes to use; it comes with a different colourful outlook. Here the purple colour was chosen. Then one clicks on finish; and thereafter, the screen changed immediately to purple, as seen Figure: 4.19.

![Theme change option](image)

Figure 4.19: Theme change option
4.5.3 MAKING THE APPLICATION THE DEFAULT SMS

Here, the default messaging app was changed to UJSMS antivirus App - for the purpose of the test, without making the UJSMS antivirus the default-messaging App, the App will not work with the phone, as expected. Figure: 4.20 presents the pictorial layout.

![UJSMS default-messaging app](image)

Figure 4.20: UJSMS default-messaging app

4.5.4 VIEWING THE SETTINGS AND THE SPAM

After the application has successfully installed on the mobile phone; the user can view the settings; and change to any setting of their choice. The user can also view the spam box. Figure 4.21 presents the picture.
4.5.5 VIEWING THE SPAM INBOX OF THE APP

This is the spam inbox; the user has the opportunity to view the spam inbox. This is where all the blocked SMS’S will be stored after the user has blocked them. Any number that has been blocked cannot send an SMS to the mobile phone any longer. Figure 4.22 presents a view of the spam inbox.
4.5.6 SMS SHOWS IN THE NOTIFICATION BAR

After successfully loading the application to the phone, a phone number, which is not in the contact list of the mobile phone was used to send an SMS. The SMS came in quite alright; and it shows at the notification bar, notifying the user about the incoming SMS, as presented in Figure 4.23.

![Figure 4.23: SMS showing in the notification bar](image)

4.5.7 ALLOW OR BLOCK FEATURE OF THE APP

After the SMS has been notified in the notification bar, indicating that there is a new incoming SMS; the user clicks on the message; then it brings up the display options: to block or allow? If allowed, it goes immediately to the inbox messaging; otherwise, it gets blocked. As presented in Figure 4.24.

![Figure 4.24: The allowed, or the blocked option](image)
4.5.8 SMS DISPLAYED INBOX AFTER CHOOSING THE ‘ALLOWED’ OPTION

After the user selected the allow option, the SMS goes to the mobile phone-inbox, messaging automatically without any interference; and any subsequent SMS from this particular number would henceforth be allowed into the mobile phone inbox-messaging system. Figure 4.25 shows the picture of the demonstration performed in which the ‘allow’ option was chosen.

![Figure 4.25: Showing the SMS inbox](image)

4.5.9 CHOOSING THE ‘BLOCK’ OPTION

For this section, according to the demonstration performed; the ‘blocked’ option was chosen. When the SMS comes in at the notification bar, notifying the mobile phone user that there is an incoming SMS, the user taps on the message the App displayed: block or allow; as seen in Figure 4.14. Then, the blocked option was chosen; after which the message went to spam; and the number was blocked permanently. Figure 4.26 presents the picture.
4.5.10 TESTING THE APPLICATION WITH A KNOWN CONTACT

The application is tested with a phone number that is in the contact list of the phone; and the SMS goes to the inbox directly - without any interference or blockage. As seen in Figure 4.27: “Tope Uj Hello testing App”
4.6 SUMMARY

In Conclusion, this chapter has reported the testing and demonstrations conducted, using the UJSMS antivirus Application. The chapter contains diagrams, figures and screen shots from the various testing’s and the demonstrations conducted with the App, using different versions of Android-mobile phones. The next chapter will provide the conclusions from this study; and it will suggest some remarks to improve the application.
5.0 CHAPTER FIVE

REMARKS AND CONCLUSIONS

5.1 INTRODUCTION
This chapter provides a general summary of the work done and some remarks. The application that has been developed is a very useful tool for mobile-phone users; and everyone is encouraged to get the App and try it.

5.2 GENERAL CONCLUSIONS
The mobile phone has so much becomes part of our daily life because of the technological advances associated with the phone. The technological advances in these devices has made life so much easier for us; both to our lives ranging from our personal life to the workplace. Mobile phones makes it easier to communicate with each other. Imagine a scenario; where there are no mobile phones; where one would still need to make use of postage, which would take much longer to deliver and to get feedback. With mobile devices, work is made much easier to do. One can work on the go while reading one’s emails and responding. Video-conferencing, chatting, Skype etc. All these are made possible with mobile devices. However, there have been cases of an unwanted SMS landing in users’ phones unannounced. A lot of useful information, sensitive and confidential data have been damaged on the phone due to a high influx of SMSs, which can result in the memory crashing. Consequently, it has become imperative to develop the UJSMS antivirus App to stop an inflow of unsolicited SMSs in the user’s phone.

The application was developed in an Android studio, together with Java programming language. It works in such a way that; when installed on the mobile phone any number that is not in the contact list of the mobile phone, when an SMS comes in it will be displayed at the notification bar notifying the user there is an SMS. And when the user clicks, there would be the option to either ‘allow’ or to ‘block’. If one decided to click allow it goes to the user’s inbox. If it is blocked, it gets blocked forever; and by so doing, any unsolicited SMS would not have room in
the mobile phone; any longer and the device would maintain its normal state. The App has been tested on various versions; and the results show that it works perfectly, as expected.

5.3 TEST WITH ANDROID 4.4 VERSION

Android 4.4 version (Kitkat), which was released on October 31, 2013 with API level of 19 – 20; and the App was installed on it successfully and allowed to run. The application was used as the default-messaging App. A demonstration was made with a number which is not in the mobile phone’s contact list to send the SMS. However, the SMS entered and stayed at the notification bar, thereby notifying the user that there was a message. The message was checked and the ‘allow’ button was clicked, which allowed the message to display in the inbox. At another demonstration, the ‘blocked’ button was clicked, which blocked the message permanently. A known contact (the contact is on the contact list of the mobile phone) was also used to send an SMS; and the message got displayed in the inbox messaging, without any problem. The pictures and Figures were shown in the previous chapter.

5.4 TEST WITH ANDROID 5.0 VERSION

Android 5.0 version (lollipop) was released on November 12, 2014 with API 21 – 22; the application also was run on it and tested. It also works perfectly, as expected. A demonstration was made by using a number which is not in the contact list of the mobile phone - to send a message to the mobile phone. The message came in and it displayed at the notification bar - notifying the user, the ‘block’ option was clicked; and the message got blocked. Another demonstration was made; this time, the ‘allow’ option was clicked and the message was displayed in the inbox of the mobile phone; where user can read the message. Also testing with a known contact (the contact is on the contact list of the mobile phone) was performed. Without wasting any; time the message displayed in the inbox of the phone, indicating that the user has the contact on his/her mobile phone, which implied that it was an important message, and not an unsolicited SMS.

5.5 REMARKS

Although it has been demonstrated that the App is useful and would enhance the user’s experience; it is easy to install and to use on mobile phones; but there are some jobs that could
still be done to enhance the application and its use. The application is limited to work on some versions used by the general populace; because the range targeted has the largest population of users; while the remaining version is the latest one, which the population of users would not want to invest in; because of the high cost of the device. Therefore, it is suggested that in further work; all the versions of Android should be covered in the designing of the App -regardless of the population and the cost of the device.

5.6 CONCLUSION

It may be concluded that the application will enhance the users’ experience; and the users are hereby encouraged to install the application on their devices; to learn how to use it; tell others about the App. It is user-friendly; easy to use; and finally, any device that uses it as their default option would be safe from any unsolicited influx of messages from unknown sources, or from unwanted advertising companies.
References


38. Electronic Communications and Transaction (ECT) Act www.up.ac.za/media/shared accessed 8th March 2017


PUBLICATIONS
