University of Nairobi

SCHOOL OF COMPUTING AND INFORMATICS

Enhancing Security of Mobile Banking and Payments in Kenya

MAINA GEOFFREY MANOTI

P53/79351/2015

Supervisor

Prof. W. Okelo Odongo

12th NOVEMBER 2016

THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF THE DEGREE OF MASTER OF SCIENCE IN DISTRIBUTED COMPUTING TECHNOLOGY

DECLARATION

I Geoffrey Manoti Maina do hereby declare that this research report is entirely my own work and where there is work or contribution of other individuals, it has been properly acknowledged. This project has not been presented for a degree in any other University.

Signature: ……………………. **Date………………………**

Geoffrey Manoti Maina

(P53/79351/2015)

University of Nairobi

Prof. W. Okelo Odongo,

This project report has been submitted for examination with my approval as University supervisor.

Signature: ……………………. **Date………………………**

ACKNOWLEDGEMENT

This research would not have been a success without the help of several people. First, I would like to thank the Almighty God for this opportunity He provided me with in advancing my level of education and knowledge as a whole. I would like to thank my supervisor Prof. W. Okelo Odongo for his contribution through his dedication and prudent advice during the stages of writing this research project. I would also like to thank all my lecturers and colleagues from the School of Computing and Informatics for imparting knowledge and instilling in me the quality of appreciating research and scholarly work in all levels of education.

In a special way I would like to thank my parents Stephen Ratemo and Ruth Okioga, for providing me with the foundation of education and nurturing me up to date, so as to become an independent and productive member of the society. I give thanks to my brother Julius Okioga, friends Griffins Oringo, Emily Nyatichi and mentor Mr. Nichloas Njiru for the role they played in the development of this project. Finally, I would like to appreciate the holistic quality education provided to me by my former University, the University of Eastern Africa Baraton that moulded me to be able to reach greater heights.

Table of Contents

ABSTRACT

The onset of e-commerce has led to the use of electronic devices such as computers and portable devices like phones and tablets in carrying out online financial transactions through deposits, withdrawals and funds transfer. Mobile banking has advanced e-commerce but has experienced challenges. Financial institutions are trying to cope with the dynamic nature of technology by offering convenient services to customers at the expense of security. This research demonstrates the vulnerability of mobile banking in Kenya, to cyber attacks such as phishing, ransomware, social engineering and database attacks that have led to a rise in banking fraud. Penetration testing was done on six mobile banking applications used by Tier 1 category of banks in Kenya. From the penetration, testing it was evident that most mobile banking applications were not secure, whereby they did not adhere to the Open Web Application Security Project 2013 (OWASP) guidelines, used for the development of secure web applications. A survey was also carried out for the collection and analysis of data, which guided the development of the model and prototype. This research provided solutions in enhancing security of mobile banking by demonstrating how security in both the application and network layers could be achieved through development of a secure M-banking application. These solutions involved the use of hashing algorithms like the Secure Hash Algorithm (SHA), encryption algorithms like Advanced Encryption Standard (AES) at the application layer. Encryption at the network layer was provided using Secure Socket Layer (SSL). The OWASP standards provided guidelines in the development of the application. Confidentiality, Integrity and Availability, which are pillars of security provided the basis of this study, where by to provide security in M-banking, issues to do with the CIA (Confidentiality, Integrity, Availability) triad needed to be addressed.

Key words: - Availability, Confidentiality, Integrity, Mobile banking, Open Web Application Security Project, Security.

LIST OF ABBREVIATIONS

ATM	Automatic Teller Machine
USSD	Unstructured Supplementary Service Data
SMS	Short Message Service
GSM	Global System for Mobile Communication
MS	Mobile Station
BTS	Base Transceiver Station
BSC	Base Station Controller
SIM	Subscriber Identity Module
PIN	Personal Identification Number
PUK	Personal Unblocking Key
TMSI	Temporary Mobile Subscriber Identity
IMSI	International Mobile Subscriber Identity
IMEI	International Mobile Equipment Identity
HLR	Home Location Register
VLR	Visitor Location Register
EIR	Equipment Identity Register
AUC	Authentication Centre
OTP	One Time Password
OWASP	Open Web Application Security Project
TLS	Transport Layer Security
SSL	Secure Socket Layer
MD5	Message Digest 5
SHA	Secure Hash Algorithm
NFC	Near Field Communication
MARA	Mobile Application Reverse Engineering and Analysis Framework
MVNO	Mobile Virtual Network Operator
HTTP/HTTPS	Hyper Text Transfer Protocol/Hyper Text Transfer Protocol Secure
3DES	Triple Data Encryption Standard
DES	Data Encryption Standard
AES	Advanced Encryption Standard

CHAPTER 1: INTRODUCTION

1.1 Background of the study

Mobile banking and payments is a service that provides a client with the ability to perform online financial transactions and payments using a mobile device such as a mobile phone or tablet. The customer has access to 24-hour services such as account balances, funds transfer, bill payments and ATM location.

Mobile banking has provided a convenient way in dealing with day to day bank transactions, similar to the traditional branch banking financial transactions in Kenya. These transactions include deposits, withdrawals and many other services in commercial banks and their branches, done physically. Mobile devices such as basic phones, smart phones and tablets are portable, in that they can be carried around and be used to access almost all bank and payment services.

According to Mbego and Phiri (2015), banking communities recognize the value of mobile banking and payments, whereby it has provided them with opportunities in reaching rural and remote areas, has created new markets, has led to innovation of products and services, access of payment systems and overcoming limitations in infrastructure.

The most common modes of mobile banking and payment transactions are through the use of:

1. Messages, which involves the use of Unstructured Supplementary Service Data (USSD) that uses the Short Message Service (SMS) platform.
2. Mobile banking applications, such as KCB Mobi by Kenya Commercial Bank, Mfukoni by Chase Bank and Eassy247 by Equity Bank.
3. Mobile enabled Internet browsers, through Wireless Application Protocol (WAP); a technical standard for accessing information over a mobile wireless network.

Basic mobile phones cannot be able to access mobile banking and payment services through WAP browsers and mobile banking applications. The phones use USSD that uses an SMS platform. USSD and SMS are also used to settle utility bill payments in Kenya, such as Kenya Power and Lighting Company bills, Nairobi water bills, digital television bills such as ZUKU, DSTV and GO TV.

Currently, smart phones use mobile banking applications mostly and internet browsers to access bank websites for financial transactions. They also support all other common functionalities of basic phones such as USSD and SMS in handling transactions.

The mobile banking and payments approach, despite its benefits has been faced with security and privacy issues. Some of the challenges include the use of SMS and USSD, which transfer data in plaintext. The use of mobile banking applications, which involves communication from a client's device via a mobile application to the bank servers, has also been faced with challenges. Mobile applications have not fully conformed to the Open Web Application Security Project requirements. Even though, USSD does not store messages in both the client's device and the server, the data still needs to be protected when in transit. Lack of data protection when in transit, exposes the data to different attacks such as man in the middle attacks.

Initially SMS were intended for subscribers to send non-sensitive messages across the open Global System for Mobile Communication (GSM) network and using it to send sensitive information has been a major source of vulnerability. Wagner (2006) maintains that encryption, mutual authentication, non-repudiation and end-to-end security as a whole, were omitted during the design of the GSM architecture, which shows the risk of transferring data in clear text.

M-banking applications have to be secure by providing privacy, confidentiality and security of data at the application layer. Mobile banking applications and their platforms have not fully adhered to the standards of a secure system of Confidentiality, Integrity and Availability mostly referred to as the CIA triad. Securing data and information in transit through a network, between a mobile device and server has also been a challenge. This involves use of protocols and cipher suites at the network layer for security purposes.

The GSM technology has addressed some of the security challenges through features that provide identification, authentication and encryption functionalities on the GSM network. Emmanuel and Jacobs (2007) say that equipment and their databases such as the authentication centre, equipment identity register, visitor and home location registers, Temporary Mobile Subscriber Identity (TMSI), and International Mobile Equipment Identity (IMEI) provide security in the GSM network. These equipment and their databases have however not succeeded in providing security of mobile banking and payment services. This is because vulnerabilities do not only exist at the network layer but also at the application layer. End to end security from the application layer to the network layer is required for M-banking, to be able to secure business transactions. It is up to the banks and financial institutions to ensure robust end-to-end security solutions for their customers, which involves protection of data when stored or in transit from the mobile device to the bank servers.

Challenges in addressing the above issues are because of conflict in offering convenient services versus security of the services. Marous (2013) suggests that customers expect convenience, as well as security. The two factors conflict as earlier stated. To improve security, multiple authentication processes are required, which reduce the convenience of services.

In this research, security vulnerabilities were identified and solutions provided. The solutions were because of development of a model, a secure mobile banking and payment application prototype, configuration of secure application and backend servers and the use of a secure network. These provided solutions in achieving end-to-end security when performing mobile banking and payment transactions, therefore addressing the issues to do with security and privacy of data.

1.2 Problem Statement

Mobile banking has experienced security and privacy issues. These security concerns are because of vulnerabilities in mobile banking applications, the use of Unstructured Supplementary Service Data and the network used for performing transactions. The current M-banking applications have not adhered to the Open Web Application Security Project guidelines used to develop secure mobile web applications. This has led to vulnerabilities when using M-banking applications during transactions. The GSM network that provides security among communication devices has also been faced with challenges related to encryption and authentication algorithms, which have been reversed engineered and broken into. This has exposed the network to cryptographic, denial of service, brute force and replay attacks. The use of USSD has also experienced vulnerabilities when handling M-banking transactions. USSD uses an SMS platform, which was not developed for transferring sensitive and confidential messages. The above issues have affected security in M-banking and an end to end secure channel needs to be provided to secure M-banking transactions when data leaves an application, when in transit and at rest.

1.3 General Objective

The general objective of this study was to demonstrate insecurity of mobile banking and payments in Kenya and provide solutions to enhance M-banking security, by demonstrating how security in both the application and network layers could be achieved for provision of end-to-end security.

1.4 Specific Objectives

1. To perform penetration testing on the current mobile banking applications used by the Tier one banks in Kenya.
2. To come up with a model that would guide the development of secure mobile banking applications.
3. To develop a mobile banking application prototype that meets the Open Web Application Security Project standards.
4. To configure an application and backend server that uses a secure network and stores client's data in a secure way.

1.5 Research Questions

This research attempted to answer several fundamental questions.

1. What was the current security state of mobile banking and payments in Kenya?
2. What were the security issues related to mobile banking and payments?
3. Which was the appropriate model for providing security in mobile banking?
4. How should we secure mobile banking and payments in both the application and network layers?

1.6 Justification and Significance of the Study

The mobile banking and payments security research would provide the following benefits to its stakeholders.

1. It would enhance security of data when stored and in transit.
2. It would demonstrate the use of cipher suites and protocols in securing data in both the application and network layers.
3. Minimize the losses incurred because of cybercrime.
4. Create stakeholders' confidence, in the use of mobile banking and payment platforms.

1.7 Scope of the study

The scope of this research was limited to mobile banking and payment applications, used by the Tier one banks in Kenya. The study, involved provision of privacy and security measures while exchanging data from a customer's mobile device to the banks server, via a secure channel.

1.8 Limitation of the study

Limitation of the project was in getting information on the current mobile banking and payments security status from the Tier one banks in Kenya, which they consider being a vulnerability to security.

The GSM network is closely related to the study and analysing the current GSM network in Kenya provided a wide scope, which was challenging. Getting to know the current GSM architecture implemented and how secure it was, in promoting security of mobile banking was the major issue.

1.9 Definition of terms

Encryption refers to the process of changing a message in readable plaintext form to a coded form referred to as a cipher text so that only authorised parties can be able to view it. (Lehtinen, Gangemi 2006).

Mobile banking is a service that provides clients with the ability to perform online financial transactions remotely using a mobile device, when they cannot access the traditional bank branches and automatic teller machines. (Nicoletti, 2014).

A vulnerability is a weakness in a system that can be exploited by an attacker, which would make him gain access to sensitive and confidential information. (Goulder, 2011).

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

The Global System for Mobile Communication (GSM) technology mostly provides security for mobile banking and payments. GSM technology incorporates security features offered by a Telecommunication Service Provider to its subscribers. According to Van der Merwe (2003), the Global System for Mobile Communications has experienced a number of challenges in providing end-to-end security as it continues to grow. These challenges need to be addressed as it is of great importance to have an end-to-end data protection mechanism in place. An-end to-end protection mechanism would involve data protection from a client's mobile device to the banks server where data is stored or rather from the front end to the back end of an application. This research covered the current GSM security features and the proposed security enhancements for end-to-end security to secure the use of M-banking in Kenya.

2.2 GSM Technology

Mobile banking and payments like other mobile services are deployed over the Global System for Mobile Communication network initially designed for voice traffic. GSM technologies have evolved starting from the 2nd Generation (2G) that uses GSM voice oriented system based on digital technology. 2G as mostly referred to, provides voice and data services. 3rd Generation (3G) provides high speed voice oriented system, integrated with data services such as General Packet Radio Service (GPRS) and Code Division Multiple Access (CDMA).

The most recent and still being deployed in Kenya is the 4thGeneration (4G). According to Kamal (2012), 4G is based on the internet protocol network, which provides voice, data and multimedia service to subscribers. In 4G technology, a mobile device connects to an Internet Protocol network and is allocated an IP address. The IP address will be used to access voice, data and multimedia services. Both 3G and 4G are found in urban areas such as city areas and major towns but also some strategic remote areas in the rural.

Subscribers are forced to revert to the old 2G network in rural or places with poor network coverage when accessing banking and payment services. 2G network is less secure when compared to the 3G and 4G networks. The above process referred to as backward compatibility provides weaker authentication and encryption schemes. GSM provides security vulnerabilities when it comes to mutual authentication, data confidentiality and non-repudiation for messaging services.

The main security issues in GSM are related to encryption and authentication algorithms. According to (Chikomo et al, 2006), the common algorithms used for encryption have been reversed engineered for instance A5 and authentication algorithms such as A3/A8 have been broken into and both the root key (Ki) and subscriber's identity revealed. When both the Ki and International Subscribers Identity (IMSI) are obtained, an attacker can use available tools to clone a subscriber's SIM card in a couple of hours. Kaur, Kaur et al, (2004) describes the various GSM network attacks, which include cryptographic attacks, denial of service, brute force and replay attacks. Kaur's suggestion on how to mitigate issues on the attacks, involves the use of new stronger encryption algorithms to replace A5/1 and A5/2 and a more cryptographically secure algorithm to replace A3.

2.3 Cellular components and their security features

The GSM architecture can be divided into three parts namely: The Mobile Station (MS), which is the mobile device and the Subscriber Identity Module (SIM) carried in it, the Base Station Subsystem made up of Base Transceiver Station (BTS), Base Station Controller (BSC), and the Network Subsystem.

The Base Transceiver Station is the main component of a geographical unit. The BTS connects a mobile subscriber to the cellular network for the purpose of transmitting and receiving information services through voice, data and multimedia services. The BSC interfaces BTSs', which are connected to it by a microwave or cable. The BSC also routes calls between the Base Transceiver Stations. Radio signals are translated by the BTS into digital format and the digital signals are then transferred to the BSC.

The SIM is a memory card integrated circuit that holds identity information of a subscriber. Stamp (2011) says that the SIM card contains the International Mobile Subscriber Identity (IMSI). The IMSI is a 15digit unique number that identifies a subscriber. Similar to the IMSI is the Temporary Mobile Subscriber Identity (TMSI) a temporary number, shorter than the IMSI assigned by the Service Provider to a phone on temporary basis. It identifies the phone and its owner in the geographical unit or the cell they are located. When the phone moves to a different cell it gets a new TMSI key.

According to Kamal (2012, p.132), both the Base Transceiver Station and the Mobile Station perform ciphering before initiating for a call or before connecting, to receive a call. This applies to other services a part from calling, like for data traffic and control data channel. This makes wireless communication secure between the MS and BTS. The security issue therefore is providing security between the mobile device and the BTS and after data leaves the BTS on its way to the application or database server. Database in the servers needs to be protected too where data is at rest.

2.3.1 Personal Identification Number (PIN)

Personal Identification Number is made up of four numbers and is stored on the SIM card of the cell phone. When a phone is turned on, the SIM checks the PIN. In case of three consecutive faulty PIN inputs, a Personal Unblocking Key (PUK) is required to unblock the device. In case of 10 faulty PUK inputs, the SIM is locked and the subscriber must ask for a new SIM. Mobile banking in Kenya mostly relies on the PIN for authentication. When a user gets a new PIN from a bank it is a common practise to change the new PIN to something different that the user will use to authenticate himself as a security measure.

2.3.2 International Mobile Equipment (IMEI)

International Mobile Equipment Identity is a 15-digit unique number that identifies each phone. The manufacturer incorporates IMEI in the cellular phone. When a phone tries to access a network, the service provider verifies the IMEI from a database. The database contains a list of stolen phone numbers. If the IMEI of a stolen phone is found in the database, the service provider denies the device connection to the network. The IMEI is located on a white sticker/label under the battery and can be displayed by typing ***#06#**.

2.3.3 PIN authentication mechanisms

According to Nyamtiga, Sam and Laizer (2013), the current mechanisms for authentication have been reverse engineered and are subjected to guessing and brute force attacks.

The use of four-digit PIN numbers produces 10,000 PIN combinations that are used by hundreds of thousands of end users. Users are certain to share the same passwords and it is more likely for one’s password to be guessed, which is made even easier by people using their years of birth for authenticating themselves.

Allowing a combination of numbers, letters and symbols referred to as hexadecimals adds more possible combinations, makes password guessing and brute force attacks hard to achieve. With this combination by using the same 4-character space we can have 10 numeral digits (0 through

9), 52 alphabetical letters (26 small letters and 26 capitals), and at least 10 symbols (! @, #, $, %, ^, &, *). This makes a total of more than 72 characters (10 + 52 + 10 = 72) which when combined in fours produces at least 26 million PINs (72^4 = 26,873,856). Brute force attacks would require more time to decipher such a PIN.

Even though it is secure, this would be complicated in terms of forgetting or blocking password after three consecutive wrong password inputs. An example would be when inputting a capital letter; you input a lowercase letter by mistake. This would be a challenge to most users therefore not a good idea. Passwords should be easy and simple but secure, therefore providing another cause of disagreement in simplicity versus security.

2.3.4 HLR, VLR and EIR registers

Home Location Register, Visitor Location Register, Equipment Register and AUC are databases used to store subscribers' details and information. Van der Merwe (2003) maintains that HLR databases contain permanent information about each subscriber such as call forwarding, call identification preferences, location activity and account status. The service provider manages HLR.

Visitor Location Register stores temporary data about a subscriber. It is located inside the Mobile Switching Centre (MSC). Emmanuel (2007) points out that when a subscriber moves to a new location, the new MSC requests the VLR from the HLR of the old MSC.

Equipment Identity Register (EIR) is a database located near the MSC. It contains information identifying cell phones. The Authentication Center (AUC) is the first security level mechanism for a GSM cellular network. The AUC is a database that stores the list of authorised subscribers of a GSM network. It is linked to the MSC and checks the identity of each user trying to connect. The AUC also provides encryption parameters to a secure cell in the network.

2.4 Securing SMS and USSD platforms

Data sent via SMS or USSD is vulnerable to interception if not well protected (United Nations Conference on Trade and Development, 2012). The SMS and USSD platforms were mainly developed for transferring non-sensitive information or data in the GSM network. The GSM network does not provide text encryption, mutual authentication between the mobile device and the banks server.

While data is in transit, a mobile network operator in a Telecommunications Service Provider can alter it and tracing the breach is not easy. This also violates the pillar of security of non-repudiation whereby in case the data is altered, tracks and logs are not available to lead us to the source of breach of security and the actor responsible for it.

In USSD, the user sends a USSD string to the banks server, after inputting a Personal Identification Number. The server sends back a message notifying the user that it is ready to accept the banks SMS message. As discussed above using the USSD, approach a mobile network operator may have full access to the banking details sent by a customer to his bank, through the banks server, which interacts with the bank's database.

In this case, we depend on the code of conduct of the mobile operator, which is a major security vulnerability by exposing the data to him, thus making mobile banking vulnerable when using USSD, which transmits data in plaintext. According to Narendiran, Rabara and Rajendran (2009), in SMS based approach for mobile banking, when a user sends a PIN number to the bank's server and the server accepts the requests, the approach is insecure because the data is transmitted in clear text and the network operator has full access of the data.

SMS spoofing is an attack that involves a masquerader altering the address field in the header of an SMS message with a different alphanumeric string. A message might appear genuine but in the real sense, a third party has tampered with it. This calls for encryption of SMS messages when being stored and in transit. Emmanuel and Jacobs (2007) suggest the use of an Advanced Encryption Standard (AES) that encrypts a message before it is transmitted.

Kaur (2012) states that an attacker, who can also execute 'SMS spoofing' by injecting messages into the network, which contain a "spoofed" originator's ID, can amend contents of a message. Currently encryption provided in the GSM network is between the Base Transceiver Station (BTS) and the Mobile Station (MS) as pointed out by Narendiran, Rabaraand Rajendran (2008). A5 is the encryption algorithm mostly provided in the GSM network. A5 has been demonstrated to be vulnerable and a more secure algorithm needs to be in place. According to Steve (2003), A5, which is one of the commonly used algorithms for encryption in the GSM system, has already been reverse engineered.

2.5 Web browsers and Mobile banking applications

The Wireless Application Protocol (WAP) used by applications and devices that use wireless communication can access mobile banking and payments. Mobile phones access M-banking services by using WAP browsers. Security threats and vulnerabilities found in mobile devices are similar to what is experienced by computers when accessing financial services, through wireless connectivity.

Mobile devices use WAP to connect to the WAP Gateway, which then connects to the banks server. Mobile browsers need to be up to date with patches that provide security when accessing websites. Use of old versions of mobile web browsers poses a security threat whereby they do not use up to date authentication and encryption protocols, and cipher suites.

From the banks server side, it would be advisable to blacklist older versions of applications and web browsers. This can be done by having a server-side filter to check for blacklisted applications such as web browsers and M-banking app versions. The user should therefore receive an error message and is asked to upgrade his web browser or application to the latest version. Through this, security is provided to both parties, the customer and bank. Narendiran, Rabara and Rajendran (2009), maintain that it is difficult to provide end-to-end security through WAP, because of data not being encrypted at the gateway during the process of switching protocols, which leads to security concerns for mobile banking in WAP.

According to Chikomo (2006), Wireless Application Protocol using Transport Layer security (TLS) and I-mode that use Secure Socket Layer (SSL) should be provided in web browsers to provide security of data when accessing bank facilities. Secure Socket Layer and Transport Layer Security are protocols that provide encryption of data used by websites. WAP websites and devices can facilitate the use of digital certificates and signatures, which are implemented by having SSL and TLS protocols in place. The TLS protocol provides security by SSL, which encrypts data while in transit in the transport layer.

The SSL protocol uses keys for encrypting and decrypting data. Public keys encrypt data while private keys decrypt data. According to Khan (1996), Kerckhoff's principle states that cryptographic systems should be secure even if everything about the system is known to the public, apart from the Key used to encrypt data. This highlights that security of a cryptosystem must depend on the Key and not security of any other part of the system, examples being the system components, how they interact and how the system functions as a whole.

One can use tools such as OWASP tools, in carrying out penetration testing to determine how secure a system is. Standards such the Open Web Application Security Project (OWASP) and Application Security Verification Standard Project OWASP (ASVS) provide criteria for developing secure applications. OWASP provides a basis for testing applications, technical security controls and their environment. OWASP also provides web security, application security, vulnerability assessment of web-developed applications such as web browsers and M-banking applications. Applications developed should meet the standards dictated by OWASP.

2.6 Network layer

Use of mobile banking and payments in Kenya mostly relies in the use of the GSM network, through Telecommunication Service Providers such as Safaricom, BhartiAirtel and Orange. Banks such as Equity bank in Kenya through its mobile subsidiary Equitel is competing for Kenya's mobile money space in offering mobile banking and payment services.

This shows the importance of the network to financial institutions and how they complement the use of M-banking and payments in Kenya, by providing this main resource to be able to perform financial transactions. Banks like Equity are trying to offer their own mobile operation services by becoming a Mobile Virtual Network Operator (MVNO). This has involved the use of technologies such as the Ultra-thin SIM cards, even though this technology has been criticized on its security level, in that it is vulnerable to risks such as PIN theft and Denial of Service attacks. This corroborates the challenge of providing security as technology advances.

According to Narendiran, Rabara and Rajendran (2008), encryption provided on the GSM network is between the Base Transceiver Station and the mobile device but not provided between BTS and the application server. The channel from the mobile device to the application server needs to be secure. This can be done by providing encryption, hashing and encapsulation security mechanisms, when data leaves the application of the mobile device to the Base Transceiver Station for routing on its way to the server. The channel between the Base Transceiver Station to the application server is the most vulnerable. This is because of vulnerabilities in the authentication mechanisms and poor management of security policies for the application server, exposure of data to the public network and of lack of end-to-end security for data being transmitted.

Encryption mechanisms need to be in place to avoid theft and interference of data when in transit as unintended users who might be malicious may access the data. Recent literature also reveal that vulnerabilities can be mitigated by use of hashing algorithms like message digests, which are obtained by hashing message contents before they are transmitted across the network. The message digest is then attached within the sent message and at the receiving end; another digest is then generated from the received content, and compared with the attached message digest. If the two digests do not match, the receiver will know that the message's integrity has been compromised.

2.7 The Mobile Banking Application Server

According to Raj Kamal (2012), communication between mobile devices and a server is of the form of a client-server computing architecture. The server has larger resources and computing power than the mobile device, where by the mobile device functions as a client because of constraint of resources. The difference in resources and computing power is what creates the client-server relationship.

The application server listens on the network for incoming requests. The server then decodes the request into a format that can be understood. A designed protocol is then used to perform security checks on the received request before executing it. The application server can be located and administered by a third party application vendor or by the bank.

The server also initiates communication with the banking system that contains a database that has all the banking and security details of the users. Customer's banking details, security details, and other related user information, are kept in the database, which is integrated into the banking system. The application on the server makes use of information based on customer verification and execution of the various requested transactions. The best way to secure the application layer is by use of hashing algorithms such as Message Digest 5 (MD5) and Secure Hash Algorithm (SHA) to enforce data integrity and confidentiality, when data is stored and is in transit. These therefore secures communication between the application installed on the mobile device and the application server.

2.8 Near Field Communication (NFC)

Near Field Communication is a short-range wireless technology that provides communication between NFC enabled devices such as mobile phones, phablet, tablets, NFC enabled cards and touch points just to mention but a few. The connection and communication of the devices is usually over a distance of not less than 10 cm, frequency range of 13.56 MHZ and a data transmission of 424 Kbit/s. According to Langer and Roland (2010), NFC can be accessible even up to 20cm.

NFC is based on Radio Frequency Identification (RFID) standards. NFC provides users with the ability to perform safe, intuitive, contactless transactions, access to digital content by simply bringing devices together into close proximity. Smartphone manufacturers are equipping devices with software and hardware such as NFC antennas so that mobile devices can interact with NFC enabled devices in providing new modes of payment. Want (2011) claims that most of the top mobile phone manufacturers, manufacture top class smartphones, which are NFC, enabled.

Financial institutions are supporting this service by providing payment platforms when charging customer accounts during their day to day activities. Irrespective of the positive strokes, NFC possesses several major limitations, in that one has to have a device that contains the NFC hardware and software support, which may not be compatible with other devices. Loss of NFC enabled devices may also lead into personal information being accessed and compromised. Mobile banking and payments which uses the client-server architecture provides a better avenue in solving some of the limitations that are experienced by use of NFC and can be integrated by NFC to provide better mobile commerce services.

Dahlberg et al (2007) say that in Europe, several mobile banking and payment companies have failed and others have been discontinued in implementing the use of the NFC technology. Australia, Spain and Scandinavian countries are among the few that have been successful. In Asia mobile payment services such as Mobile Suica, Edy, Moneta, Octopus and GCash have been successful in countries like Japan and South Korea. The difference between the successful implementations of mobile banking and payments in Asia and failure in Europe is because of the culture of payment, which is tied to a specific region or country. This research looked at the NFC technology briefly and how it would be integrated with a mobile banking application in improving security and convenience of services.

Hayat (2009) suggests that financial institutions should provide adequate protection for its consumers. The protection can be provided by ensuring a stable economy, provision of interoperability of electronic system, guaranteeing security of transactions, having anti-money laundering schemes in place, and through the Know-Your-Customer principle. Knowing what stakeholders want, their payment behaviour increases the adoption of M-banking and payments.

2.9 Conceptual Framework

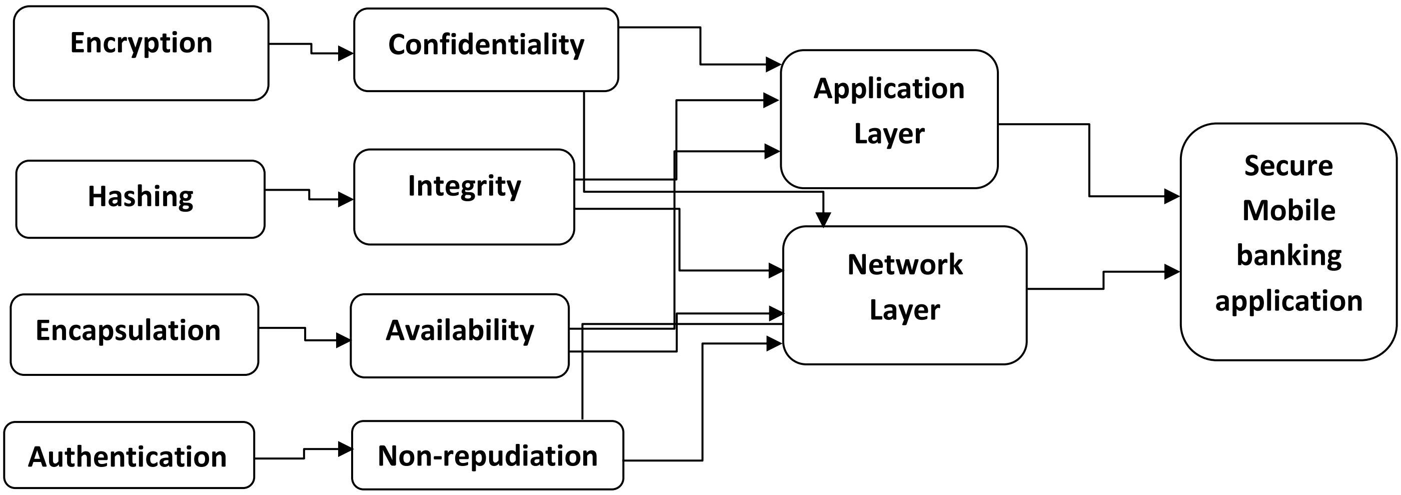


Figure 1: Conceptual framework

The conceptual framework shows the relationship between the independent variables and the dependent variable. The independent variables, which as are some of the techniques used for securing data and information, included hashing, encryption, authentication and encapsulation, having the dependent variable as the end product, a secure mobile banking application. When the strongest techniques are employed the more secure the mobile banking application.

Encryption techniques can be used to provide confidentiality as a pillar of security in both the application and network layers. Some of the encryption algorithms that provide data encryption include Data Encryption Standard (DES), Triple Data Encryption Standard (3DES) Advanced

Encryption Standard (AES) and RSA's RC2 and RC4. AES has proven to be more secure compared to other encryption schemes.

The different encryption schemes have different values in terms of the number of bits they use to encrypt information, which is in plaintext. DES provides 64-bit key size encryption, 3 DES provides 56, 112, 168 key bit sizes and AES provides 128, 192 and 256 key bit sizes. The higher the key bit size the more secure an encryption algorithm is. This is why AES is considered a more secure algorithm than 3DES or DES, where by it produces a maximum 256-bit size compared to the maximum key bit size of 168 in 3DES and a maximum of 64-bit size in DES. These values can be used as the independent variable values, which determine the output, in this case the level of security for the mobile banking application.

The same would apply to hashing algorithms like MD5 or SHA. MD5 has 128 bits, SHA1 256 bits and SHA2 224, 256, 384, 512 bits. The higher the hashing bit values the more secure the hash would be, in that hashing being an independent variable, the more hash bits of the data in plaintext the more secure the application would be. SHA2 is therefore more secure when compared to SHA1 and MD5. Devices communicating select the most secure hash or encryption algorithm and the algorithm they both support. This determines the security of an application.

It is secure to have a system that provides two factor or multiple factor authentication schemes in both the application and network layers. For example, in the application layer a system that provides multiple factor authentication should provide an authentication mechanism that offers security based on something an individual knows, is and has to pass the authentication stage. The more authentication schemes for an application the more secure the application.

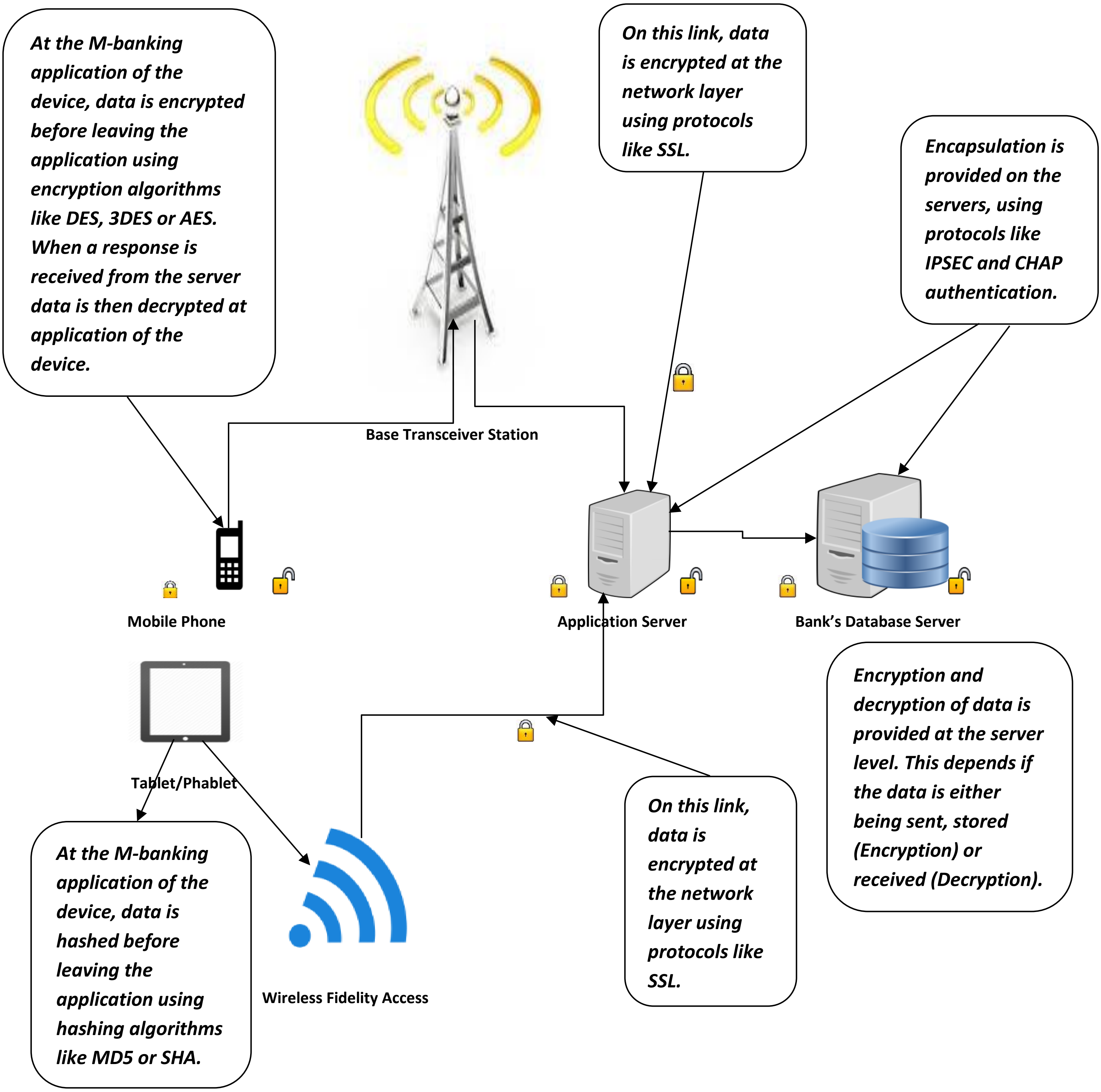
According to Konheim (2007), message digest algorithms, possibly RSA's MD5 and NIST's SHA provide the hashing of data, where data is hashed and is not readable in plain text. This provides confidentiality and integrity of data at the application layer. Confidentiality and integrity of data is provided at the network layer using a Secure Socket Layer (SSL). SSL uses digital signatures to provide secrecy and authentication. The Message Authentication Code (MAC) provides authentication in the SSL protocol.

Encapsulation of data in both the application and network layers provides availability of a system. This is by shielding the inner content of data in the network layer and providing very limited information of the inner contents of an application and the system as a whole. This should not deny the knowledge of how a system has been developed and how it works, according to Kerckhoff's principle as earlier mentioned. The table below shows some of the security problems and proposed solutions in mobile and wireless computing systems.

Security Problems	**Description of the problem**
Confidentiality	Only destined users must be able to read data. (Encryption of data before transmission and deciphering data at the users end is a method employed for ensuring confidentiality).
Integrity	It avoids a user from receiving messages that have been interfered with, which raises flags that data has been corrupted or compromised.
Availability	A systems hardware and software not being available because of technical and security issues affects the availability of a system. A system should be **99.9%** available/accessible with a downtime of **0.1%.**
Non-repudiation	A sender cannot deny having sent a message. An example, if a user performs a bank transaction, he should not be able to deny.

Table 1: Security Pillars

2.10 Conceptual Architecture



At the M-banking application of the device, data is encrypted before leaving the application using encryption algorithms like DES, 3DES or AES. When a response is received from the server data is then decrypted at application of the device.

On this link, data is encrypted at the network layer using protocols like SSL.

Encapsulation is provided on the servers, using protocols like IPSEC and CHAP authentication.

Encryption and decryption of data is provided at the server level. This depends if the data is either being sent, stored (Encryption) or received (Decryption).

On this link, data is encrypted at the network layer using protocols like SSL.

At the M-banking application of the device, data is hashed before leaving the application using hashing algorithms like MD5 or SHA.

The conceptual architecture demonstrated the relationship during communication, when a mobile device, transmits data to the bank servers and when data is sent back to the device from the bank servers.

Currently security is provided between the base transceiver station and the mobile device. Security for mobile banking and payments may be enhanced by providing security at the application layer when data leaves the device and at the transport layer when data is in transit.

Data can be secured at the application layer by hashing algorithms, possibly MD5 or SHA, and encrypting data with encryption algorithms like DES, 3DES and AES as discussed earlier, and the same can be at the network layers via protocols like SSL. When data is at rest, it should also be secured. Data can be secured when at rest by providing authentication, encryption and encapsulation at the database level.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Research Design

The exploratory research design was used in this research. The design involved addressing the research problem related to the topic in order to understand the topic, objectives that were to be achieved and research questions to be answered.

This form of design assisted in thorough investigation in seeking answers and explanations by exploring different areas and variables that affected our output and topic. This research was a quantitative type of research. The research dealt mostly with numerical data to describe our independent variables as discussed earlier and examine their relationships in determining the output.

The research design also dictated how data was to be collected and analysed to achieve the objectives of the research and use of resources efficiently and effectively to get relevant information related to our research problem with minimum use of time, money and effort. This made the study viable. The exploratory research design is flexible. This enabled us to use tools and techniques like penetration testing to collect and analyse our data and information.

3.2 Data Sources

The primary sources of data collection were the current M-banking applications of the Tier 1 banks in Kenya, which provided data by running penetration testing on them. Mobile Application Reverse Engineering and Analysis Framework (MARA) and Charles Proxy tools were used to perform penetration testing. The guidelines on the penetration testing were provided by OWASP (2013) standards.

Secondly, stakeholders who interact with M-banking and payment solutions, through bank transactions and other forms of utility payments also provided data for the research by participating in a survey.

3.3 Tools and Techniques

One of the tools used to perform penetration testing was the **Mobile Application Reverse Engineering and Analysis Framework** (MARA). MARA puts together commonly used mobile application reverse engineering and analysis tools, in testing mobile applications against the Open Web Application Security Project (OWASP) mobile security threats and vulnerabilities.

Pen testing was done on android applications, which use the android application package file format **.apk**. The mobile banking and payment applications use this file format and extension, android being an operating system built on the Linux kernel.

The penetration testing involved the following processes:

1. **APK Reverse engineering**

- Disassembling Dalvik bytecode to smali bytecode via baksmali and apktool
- Disassembling Dalvik bytecode to java bytecode via enjarify
- Decompiling APK to Java source code via jadx

2. **Preliminary Analysis**

- Parsing smali files for analysis via smalisca
- Dump apk assets, libraries and resources
- Extracting certificate data via openssl
- Extract resource strings and app permissions via aapt
- Identify methods and classes via ClassyShark
- Scan for apk vulnerabilities via androbugs
- Analyze apk for potential malicious behaviour via androwarn
- Identify compilers, packers and obfuscators via APKiD
- Extract execution paths, IP addresses, URL, URI, emails via regex parsing
- Domain SSL scan via pyssltest and testssl

The most resourceful process under the preliminary analysis was scanning of the .apk files for vulnerabilities via androbugs. Androbugs dumped the analysis, inform of a report on the Androbugs folder. An example of the report can be obtained on the Appendix 1 section.

The report provided information about the application in terms of its security features and vulnerability assessment, which were presented as warning, information and notice flags based on the **OWASP** standards. The ***red flags*** represented ***critical security*** vulnerability status that needed immediate response, the ***yellow flags*** represented ***mild security*** vulnerabilities and the ***purple flags*** represented ***positive status*** or ***recommendations*** for the application.

Security features and vulnerability assessment were mostly on the application and network layers of the web application, assuming a mobile device was communicating with a server through the web. The generated report gave recommendations on the applications by providing credible website references on how the application vulnerabilities could be fixed.

Penetration testing was also done by using a third party software **Charles**, which is a web proxy (HTTP Proxy / HTTP Monitor) that runs on a computer. A certificate with the file extension **.pem** a file format used for certificates by OpenSSL and other SSL toolkits, was first obtained from the computer by Charles and then installed on the mobile phone on the settings panel under the security section.

The computer was then configured as a proxy for the mobile device, making them belong to the same network. The proxy settings were inputted on the wireless Service Set Identifier (SSID) connection of the mobile device, having the proxy IP address, as the address of the computer, which then established a connection between the two. Data was then collected through HTTP(S) Proxying, when the mobile banking application accessed the internet through the proxy. The data was then analysed. Figure 10 and 11 on the Appendix section 2 displays information obtained from Charles, where a user's password or **PIN** was displayed in plain text, which is a major security vulnerability.

According to the OWASP Top 10 application security risks list (2013), it shows a list of flaws that are severe and prevalent in providing security of web applications. Software especially web applications are required to conform to the OWASP Top 10, 2013 standards to achieve the required standards during development and before release of mobile applications to the market. The list below shows web application threats.

1. **A1-injection**- It includes injection flaws such as SQL, OS and LDAP injection, whereby an attacker tricks the interpreter in executing commands that access data without proper authorization.

2. **A2-Broken Authentication and Session Management**- It involves incorrect implementation of authentication and session management that allows attackers to compromise an application.

3. **A3-Cross-Site Scripting (XSS)** - XSS flaws happen when applications take untrusted data and send them to a web browser without proper authorization and validation. This can happen when an attacker executes scripts in a victim's browser, which hijacks a user's session, deface web sites, or redirects the user to malicious sites.

4. **A4-Insecure Direct Object References** - It happens when a developer exposes a reference to a file, directory or database key. Attackers can manipulate files, directories and database keys to access unauthorized data.

5. **A5-Security Misconfiguration** - Having secure configurations in the application, frameworks, and servers. Default security features should not be maintained in configurations of software and hardware. Additionally, software should be kept up to date.

6. **A6-Sensitive Data Exposure** - Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify data that is not well protected. Protection such as encryption at rest or in transit as well as special precautions when data is being exchanged with the browser need to be implemented.

7. **A7-Missing Function Level Access Control** - Most applications verify function level access rights before making the functionality visible in the user interface. Access controls need to be performed on the servers when each function is accessed. Attackers can forge requests that would make them access functionalities without proper authorization.

8. **A8-Cross-Site Request Forgery (CSRF)** - It happens when an attacker forces a log-on on the victim's browser then sending a forged HTTP request, which includes the victim's session cookie and other authentication information, that are vulnerable to web applications.

9. **A9-Using Components with Known Vulnerabilities** - Components, such as libraries, frameworks, and other software modules run with full privileges. If these components are exploited this may lead to loss of data and severe takeover.

10. **A10-Unvalidated Redirects and Forwards** -Web applications frequently redirect and forward users to other pages and websites. Without proper validation, attackers can redirect victims to phishing or malware sites.

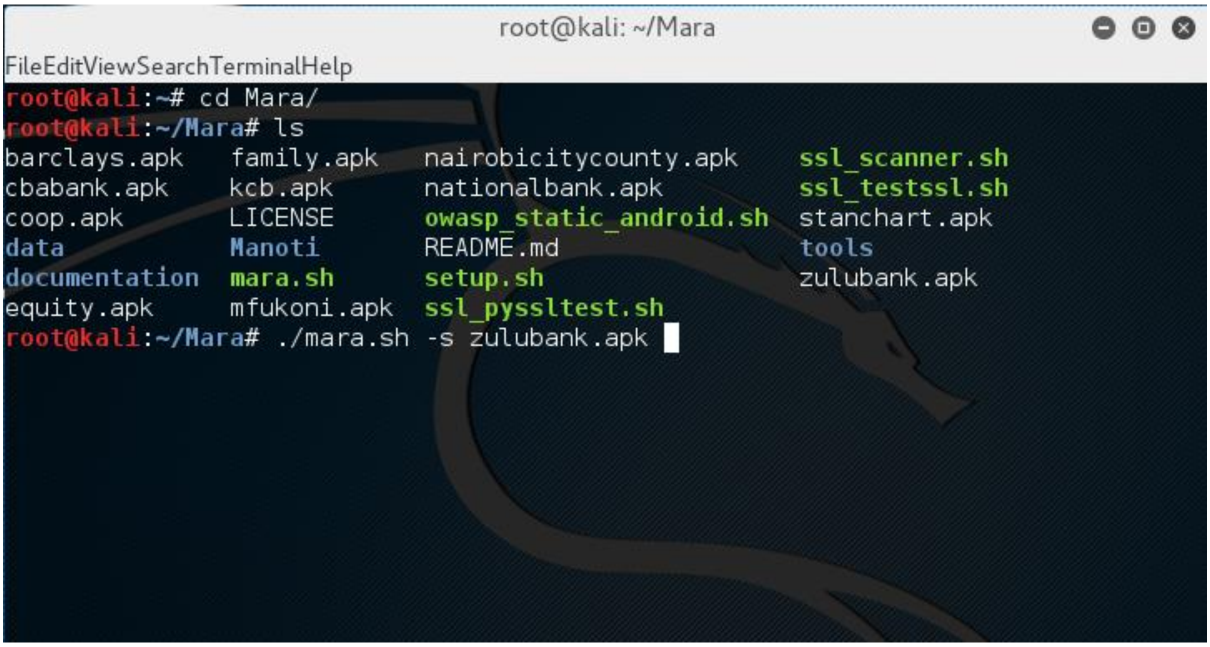
CHAPTER 4: DATA COLLECTION AND ANALYSIS

4.1 Penetration Testing

Data was collected and analysed by running application assessment on mobile banking and payment applications. The analysis was performed on six distinct mobile banking applications used by the tier one category of banks. The analysis involved a combination of vulnerability scanning, code review, and most important of all penetration testing.

The penetration testing involved the process of first installing Kali Linux, Operating System (OS). The operating system provided the penetration testing tool MARA with the platform for its operation. After installation of the OS, updates and patches for the operating system were then instaled from the linux public repository to update the libraries that MARA used as prerequisites for its installation and operation. MARA was then installed on the computer.

Mobile banking and payment applications were downloaded on the mobile phone via the google playstore website, the official website for downloading android mobile applications. The applications were then transferred to the computer via a universal serial bus datacable. A folder with the name Mara was then created and the mobile banking applications were then dumped on the folder. The figure below shows the different M-banking applications with the extension .apk dumped on the Mara folder.

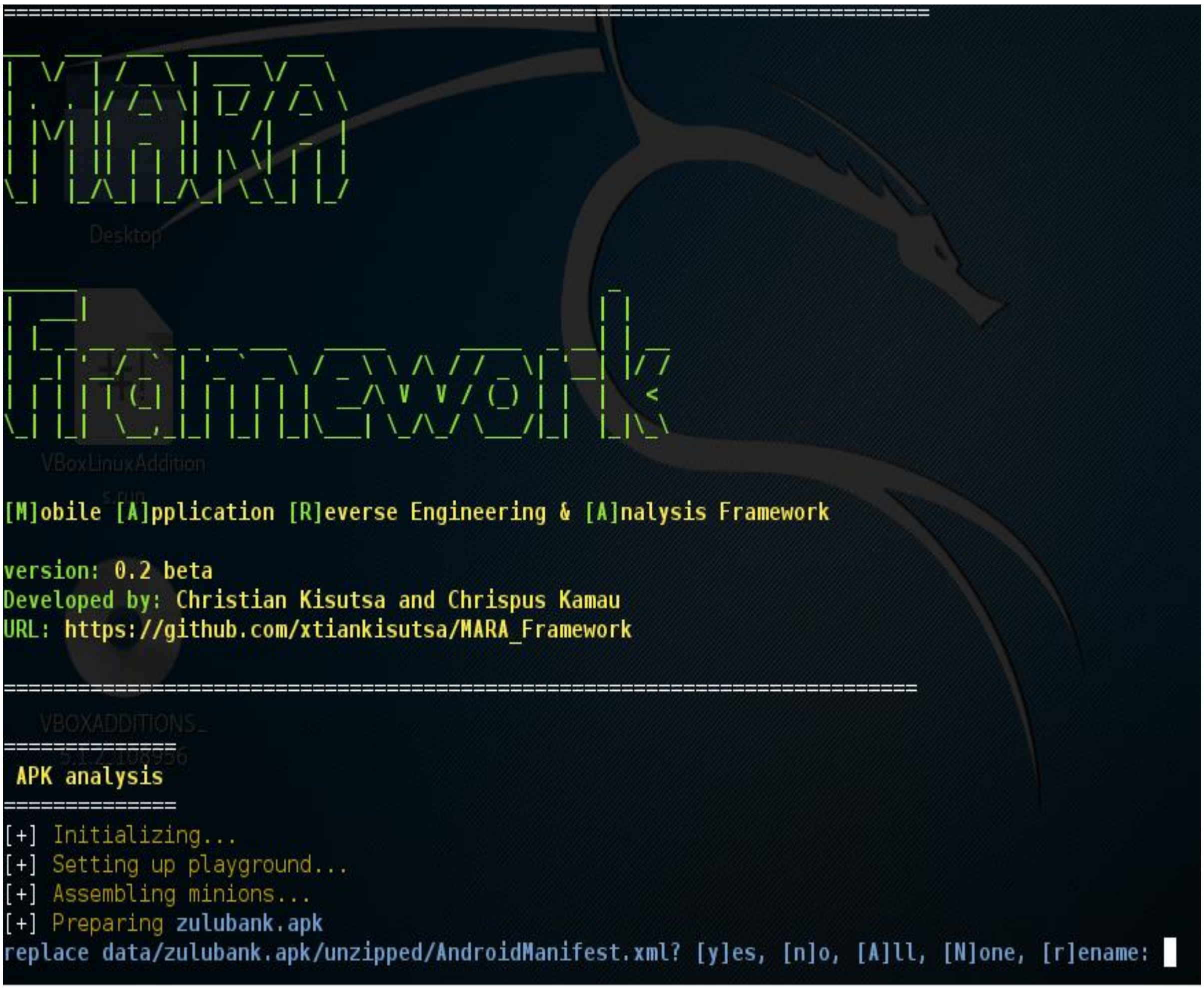


```
root@kali:~# cd Mara/
root@kali:~/Mara# ls
barclays.apk    family.apk    nairobicitycounty.apk    ssl_scanner.sh
cbabank.apk     kcb.apk       nationalbank.apk         ssl_testssl.sh
coop.apk        LICENSE       owasp_static_android.sh  stanchart.apk
data            Manoti        README.md                tools
documentation   mara.sh       setup.sh                 zulubank.apk
equity.apk      mfukoni.apk   ssl_pyssltest.sh
root@kali:~/Mara# ./mara.sh -s zulubank.apk
```

Figure 2: Pen testing 1

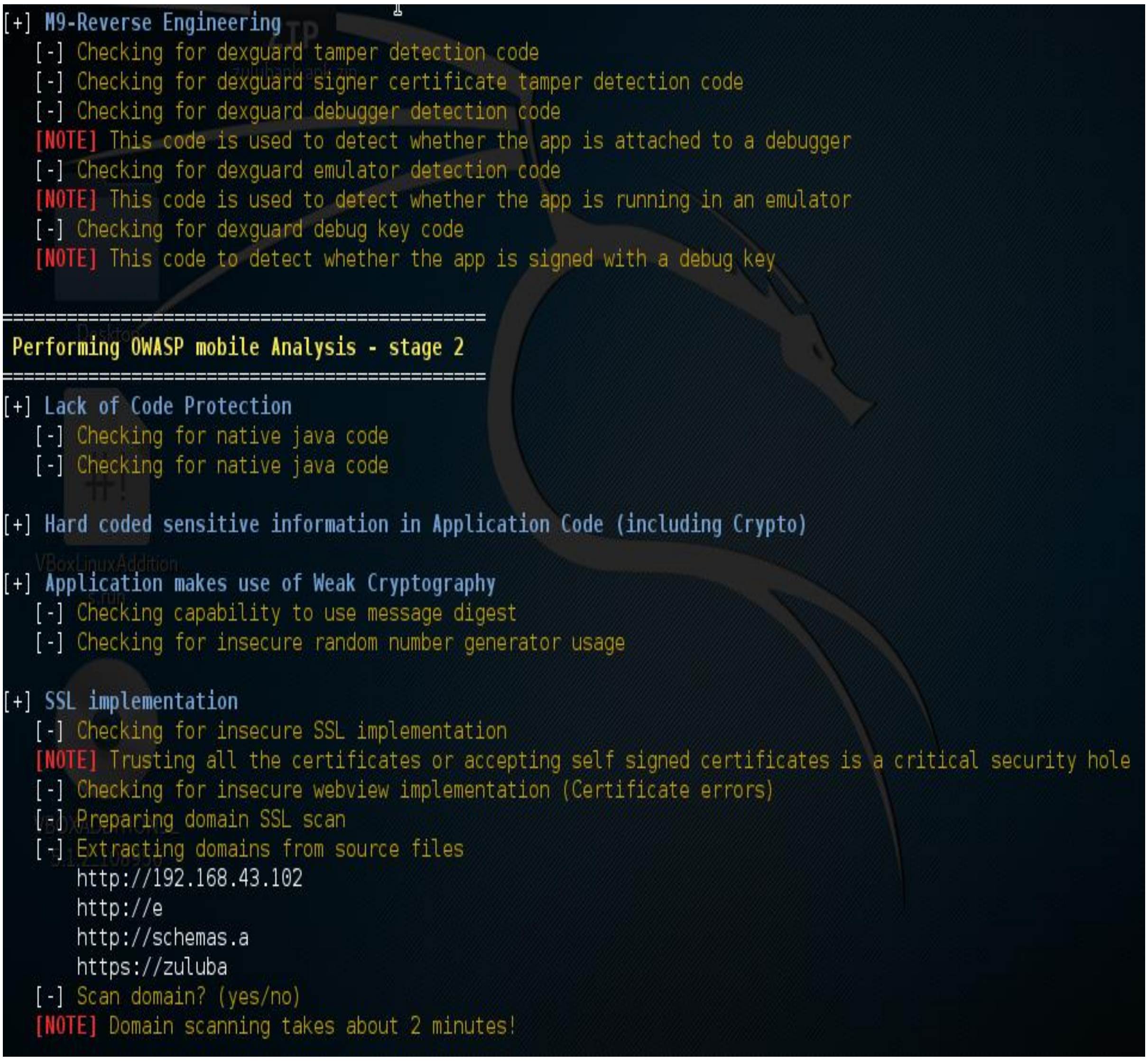
One would then navigate into to the Mara folder, which contained the different applications and then run the command **./mara.sh –s** with the name of the application to be analysed such as **zulubank.apk** and the application would then be analysed. i.e. **./mara.sh –s zulubank.apk**

The applications were then analysed one by one and reports created and dumped in their respective application folders. The figures below show the analysis process.



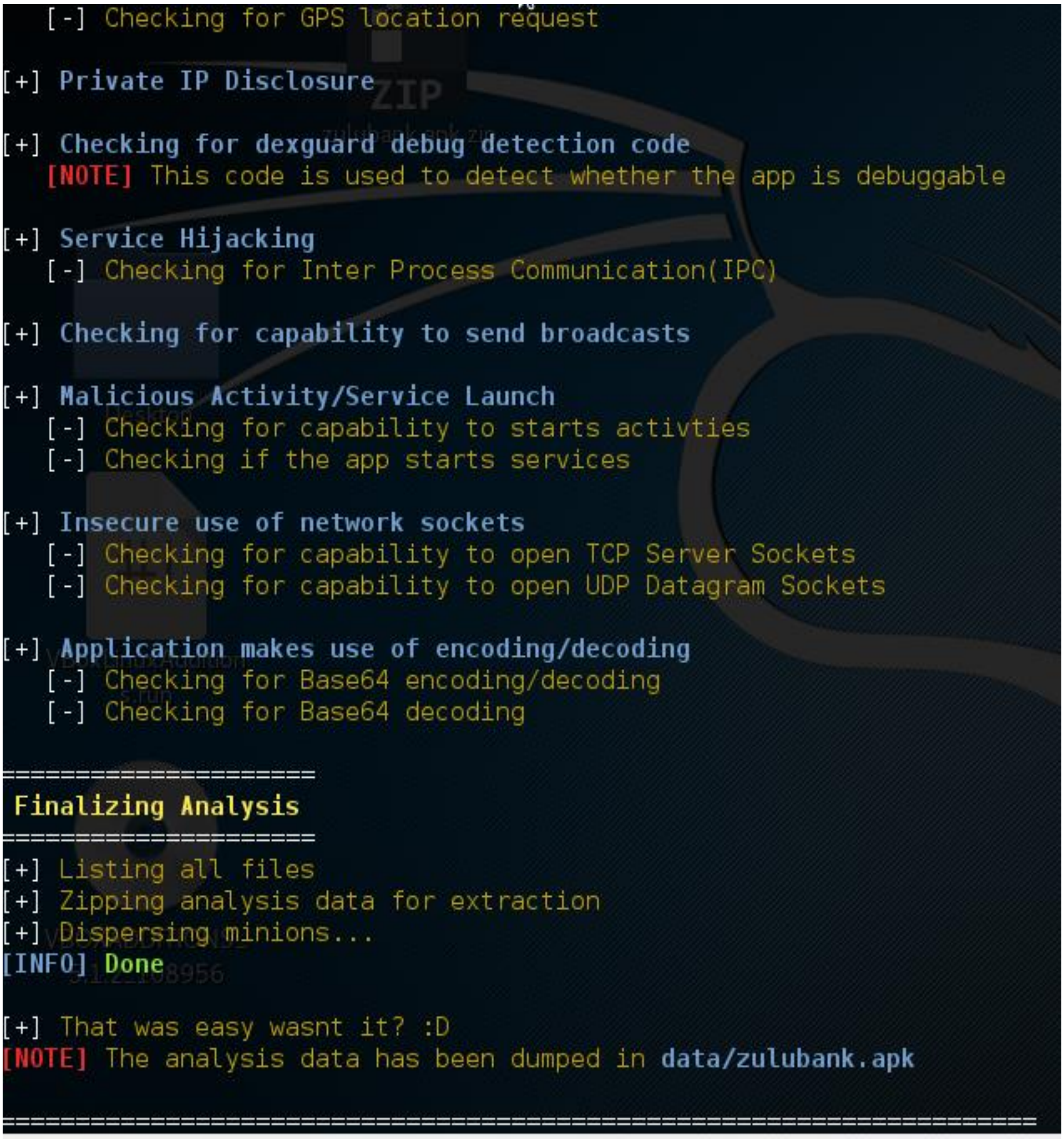
```
[M]obile [A]pplication [R]everse Engineering & [A]nalysis Framework

version: 0.2 beta
Developed by: Christian Kisutsa and Chrispus Kamau
URL: https://github.com/xtiankisutsa/MARA_Framework

==========================================================================

==============
 APK analysis
==============
[+] Initializing...
[+] Setting up playground...
[+] Assembling minions...
[+] Preparing zulubank.apk
replace data/zulubank.apk/unzipped/AndroidManifest.xml? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

Figure 3: Pen testing 2



```
[+] M9-Reverse Engineering
   [-] Checking for dexguard tamper detection code
   [-] Checking for dexguard signer certificate tamper detection code
   [-] Checking for dexguard debugger detection code
   [NOTE] This code is used to detect whether the app is attached to a debugger
   [-] Checking for dexguard emulator detection code
   [NOTE] This code is used to detect whether the app is running in an emulator
   [-] Checking for dexguard debug key code
   [NOTE] This code to detect whether the app is signed with a debug key

=============================================
 Performing OWASP mobile Analysis - stage 2
=============================================
[+] Lack of Code Protection
   [-] Checking for native java code
   [-] Checking for native java code

[+] Hard coded sensitive information in Application Code (including Crypto)

[+] Application makes use of Weak Cryptography
   [-] Checking capability to use message digest
   [-] Checking for insecure random number generator usage

[+] SSL implementation
   [-] Checking for insecure SSL implementation
   [NOTE] Trusting all the certificates or accepting self signed certificates is a critical security hole
   [-] Checking for insecure webview implementation (Certificate errors)
   [-] Preparing domain SSL scan
   [-] Extracting domains from source files
       http://192.168.43.102
       http://e
       http://schemas.a
       https://zuluba
   [-] Scan domain? (yes/no)
   [NOTE] Domain scanning takes about 2 minutes!
```

Figure 4: Pen testing 3



```
   [-] Checking for GPS location request

[+] Private IP Disclosure

[+] Checking for dexguard debug detection code
   [NOTE] This code is used to detect whether the app is debuggable

[+] Service Hijacking
   [-] Checking for Inter Process Communication(IPC)

[+] Checking for capability to send broadcasts

[+] Malicious Activity/Service Launch
   [-] Checking for capability to starts activties
   [-] Checking if the app starts services

[+] Insecure use of network sockets
   [-] Checking for capability to open TCP Server Sockets
   [-] Checking for capability to open UDP Datagram Sockets

[+] Application makes use of encoding/decoding
   [-] Checking for Base64 encoding/decoding
   [-] Checking for Base64 decoding

=====================
 Finalizing Analysis
=====================
[+] Listing all files
[+] Zipping analysis data for extraction
[+] Dispersing minions...
[INFO] Done

[+] That was easy wasnt it? :D
[NOTE] The analysis data has been dumped in data/zulubank.apk

=====================================================================
```

Figure 5: Pen testing 4

4.2 Survey

A system may be secure but can be compromised because of human error, therefore creating a vulnerability. Attacks like social engineering are difficult to combat. A person's ability to make a wrong decision and reveal his private information to an attacker, who tricks him of being a trustworthy party, in that he would like to provide assistance by confirming or updating a customer's information, is an example of social engineering attack. The customer would then end up revealing his private information to a malicious actor. This remains a major challenge in securing e-commerce as a whole. Having inadequate security background and lack of awareness on the different forms of cyber-attacks and areas of vulnerabilities, creates a vast area of exploitation by attackers.

A survey was conducted to collect and analyse information based on the security of the current mobile banking and payment applications, customers' level of awareness and perception of security. The survey provided information that was used in the development of the model by providing concepts that enabled us achieve our objectives and answer research questions of this study.

The survey involved the use of questionnaires, which were filled by 50 respondents. A sample of the questionnaire is available at the Appendix section 3. The data was then collected and analysed as shown on the next page.

Frequency Table

Using M-banking is not secure.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly Disagree	7	14.0	14.3	14.3
	Disagree	23	46.0	46.9	61.2
	Don't know	6	12.0	12.2	73.5
	Agree	12	24.0	24.5	98.0
	Strongly agree	1	2.0	2.0	100.0
	Total	49	98.0	100.0	
Missing	99.00	1	2.0		
Total		50	100.0		

Table 2: Security Q1

The figure above shows the distribution and percentage of the respondents based on their response, if using mobile banking was not secure. **14%** strongly disagreed, **46%** disagreed, **12%** did not know what to respond, **24%** agreed and **2%** strongly agreed with the statement.

Security of M-banking is wanting.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly Disagree	6	12.0	12.2	12.2
	Disagree	15	30.0	30.6	42.9
	Don't know	6	12.0	12.2	55.1
	Agree	13	26.0	26.5	81.6
	Strongly Agree	9	18.0	18.4	100.0
	Total	49	98.0	100.0	
Missing	99.00	1	2.0		
Total		50	100.0		

Table 3: Security Q2

The figure above shows the distribution and percentage of the respondents based on their response, if security of mobile banking was wanting. **12%** strongly disagreed, **30%** disagreed, **12%** did not know what to respond, **26%** agreed and **18%** strongly agreed with the statement.

Is mobile banking a security concern?

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly Disagree	1	2.0	2.0	2.0
	Disagree	7	14.0	14.3	16.3
	Don't know	2	4.0	4.1	20.4
	Agree	17	34.0	34.7	55.1
	Strongly Agree	22	44.0	44.9	100.0
	Total	49	98.0	100.0	
Missing	99.00	1	2.0		
Total		50	100.0		

Table 4: Security Q3

The figure above shows the distribution and percentage of the respondents based on their response, if mobile banking was a security concern. **2%** strongly disagreed, **14%** disagreed, **4%** did not know what to respond, **34%** agreed and **44%** strongly agreed with the statement.

Banks do not have the ability in M-banking to protect my privacy.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly Disagree	9	18.0	18.4	18.4
	Disagree	13	26.0	26.5	44.9
	Don't know	16	32.0	32.7	77.6
	Agree	8	16.0	16.3	93.9
	Strongly Agree	3	6.0	6.1	100.0
	Total	49	98.0	100.0	
Missing	99.00	1	2.0		
Total		50	100.0		

Table 5: Security Q4

The figure above shows the distribution and percentage of the respondents based on their response, if banks did not have the ability in mobile banking to protect customers' privacy. **18%** strongly disagreed, **26%** disagreed, **32%** did not know what to respond, **16%** agreed and **6%** strongly agreed with the statement.

Matters of security have an influence on my using mobile banking.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly Disagree	2	4.0	4.1	4.1
	Disagree	3	6.0	6.1	10.2
	Don't know	5	10.0	10.2	20.4
	Agree	20	40.0	40.8	61.2
	Strongly Agree	19	38.0	38.8	100.0
	Total	49	98.0	100.0	
Missing	99.00	1	2.0		
Total		50	100.0		

Table 6: Security Q5

The figure above shows the distribution and percentage of the respondents based on their response, if matters of security had an influence in using mobile banking. **4%** strongly disagreed, **6%** disagreed, **10%** did not know what to respond, **40%** agreed and **38%** strongly agreed with the statement.

Frequency Table

USSD connection problem.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly Disagree	2	4.0	4.3	4.3
	Disagree	10	20.0	21.3	25.5
	Don't know	6	12.0	12.8	38.3
	Agree	26	52.0	55.3	93.6
	Strongly Agree	3	6.0	6.4	100.0
	Total	47	94.0	100.0	
Missing	99.00	3	6.0		
Total		50	100.0		

Table 7: Issue 1

The figure above shows the distribution and percentage of the respondents based on their response, if they experienced issues in accessing M-banking services through **USSD** connection. **4%** strongly disagreed, **20%** disagreed, **12%** did not know what to respond, **52%** agreed and **6%** strongly agreed with the statement.

M-banking application connection problem.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly Disagree	4	8.0	8.5	8.5
	Disagree	15	30.0	31.9	40.4
	Don't know	6	12.0	12.8	53.2
	Agree	19	38.0	40.4	93.6
	Strongly Agree	3	6.0	6.4	100.0
	Total	47	94.0	100.0	
Missing	99.00	3	6.0		
Total		50	100.0		

Table 8: Issue 2

The figure above shows the distribution and percentage of the respondents based on their response, if they experienced issues in accessing M-banking services because of mobile banking apps connectivity issues. **8%** strongly disagreed, **30%** disagreed, **12%** did not know what to respond, **38%** agreed and **6%** strongly agreed with the statement.

Lack of enough airtime affects transactions.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly Disagree	3	6.0	6.4	6.4
	Disagree	12	24.0	25.5	31.9
	Don't know	3	6.0	6.4	38.3
	Agree	21	42.0	44.7	83.0
	Strongly Agree	8	16.0	17.0	100.0
	Total	47	94.0	100.0	
Missing	99.00	3	6.0		
Total		50	100.0		

Table 9: Issue 3

The figure above shows the distribution and percentage of the respondents based on their response, if they experienced issues of not having enough airtime that affected their mobile banking transactions. **6%** strongly disagreed, **24%** disagreed, **6%** did not know what to respond, **42%** agreed and **16%** strongly agreed with the statement.

Network problem while trying to connect to the M-banking service.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly Disagree	3	6.0	6.4	6.4
	Disagree	12	24.0	25.5	31.9
	Don't know	4	8.0	8.5	40.4
	Agree	21	42.0	44.7	85.1
	Strongly Agree	7	14.0	14.9	100.0
	Total	47	94.0	100.0	
Missing	99.00	3	6.0		
Total		50	100.0		

Table 10: Issue 4

The figure above shows the distribution and percentage of the respondents based on their response, if they experienced network connectivity issues when connecting to mobile banking services. **6%** strongly disagreed, **24%** disagreed, **8%** did not know what to respond, **42%** agreed and **14%** strongly agreed with the statement.

Mobile banking application errors.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly Disagree	4	8.0	8.9	8.9
	Disagree	10	20.0	22.2	31.1
	Don't know	8	16.0	17.8	48.9
	Agree	18	36.0	40.0	88.9
	Strongly Agree	5	10.0	11.1	100.0
	Total	45	90.0	100.0	
Missing	99.00	5	10.0		
Total		50	100.0		

Table 11: Issue 5

The figure above shows the distribution and percentage of the respondents based on their response, if they experienced mobile banking application errors such as bugs in the application. **8%** strongly disagreed, **20%** disagreed, **16%** did not know what to respond, **36%** agreed and **10%** strongly agreed with the statement.

Delay experienced when performing bank transactions.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Strongly Disagree	6	12.0	12.8	12.8
	Disagree	13	26.0	27.7	40.4
	Don't know	4	8.0	8.5	48.9
	Agree	15	30.0	31.9	80.9
	Strongly Agree	9	18.0	19.1	100.0
	Total	47	94.0	100.0	
Missing	99.00	3	6.0		
Total		50	100.0		

Table 12: Issue 6

The figure above shows the distribution and percentage of the respondents based on their response, if they experienced delays when performing bank transactions. **12%** strongly disagreed, **26%** disagreed, **8%** did not know what to respond, **30%** agreed and **18%** strongly agreed with the statement.

CHAPTER 5: RESULTS AND DISCUSSIONS

5.1 Penetration testing results

Information collected and analysed after performing penetration testing with **MARA** and **Charles** tools on the six mobile banking applications, showed that most of the applications were vulnerable and susceptible to the different types of attacks. Fig.10 and 11 on the Appendix section 2 showed one of the mobile banking applications, which displayed a customer's PIN in plain text, which is a major security weakness.

Passwords displayed in plaintext can be secured using hashing algorithms like MD5 and SHA, and encryption algorithms like DES, 3DES and AES, to hash and encrypt passwords, so that they are not displayed in plaintext but in a cipher text format. Only authorised users can access data that has been encrypted, by possessing a secret or private key used to decrypt the coded message therefore making mobile banking transactions secure.

The report on the Appendix section 1 also showed one of the mobile banking applications vulnerable areas in both application and network layers. The application did not conform to the OWASP standards where critical and mild warning flags were reported.

On Figure 12 on the Appendix section 2, it shows a code that can be executed on one of the mobile banking applications. The result of executing the code, created fake user accounts, which generated fake details such as the first name, last name, phone number, pin number and e-mail address. Continuous creation of these accounts on the servers can be resource intensive by consuming the Central Processing Unit (CPU), Random Access Memory (RAM), the hard disc space and a result of denial of service attack, where by response and availability of the mobile banking and payment services from the servers, to clients are either slow or not available.

5.2 Survey results

Information collected from the tables showed the respondents level of awareness, perception, use and adoption of mobile banking and payments. The main characteristics of a secure system based on the survey were security, reliability and availability.

A good system should be secure where by data is transmitted and stored safely. A system should also be reliable by being convenient and dependable. Availability of a system is measured based on its uptime and downtime. A good system should have an uptime rate of **99.9%** and downtime rate of **0.1%**, this would make system services available most of the time and few downtime hours, unless during scheduled maintenance hours.

From the survey, questions based on security provided information that showed respondents did not have adequate information on the security of M-banking where by the respondents believed that banks had the ability to protect their privacy. **44%** agreed with this statement, **32%** did not know their position while **16%** felt that banks did not have the ability to protect their privacy.

The respondents also believed that security was a major factor, which influenced the use of mobile banking and payments in Kenya. **78%** believed this statement was true while **10%** disagreed with the statement. The remaining **12%** did not know what to respond. This attested that security was a major pillar for provision of e-commerce services.

From the survey **60%** of the respondents agreed that M-banking was secure while the **36%** felt that M-banking was not secure. The remaining **4%** did not know what to respond. This showed the level of awareness based on security of M-banking and payments and provided information contrary to what was obtained by the penetration testing done on the M-banking applications, which showed most of the applications as being vulnerable.

From the above statistics, it is evident that the respondents are not aware of the insecurity or challenges of M-banking in Kenya. **60%** of the respondents believe that financial institutions are providing them with convenient and secure services. It is up to financial institutions to ensure that they provide convenient and secure services, even though they cannot be able to provide services completely free from vulnerabilities, because weaknesses in systems are discovered after short periods and attacks are also created and discovered after short periods of time.

Banks can reduce vulnerabilities and attacks but not prevent them completely, by ensuring applications conform to security standards such as OWASP (2013) and by conducting regular audits on their applications through their technical departments.

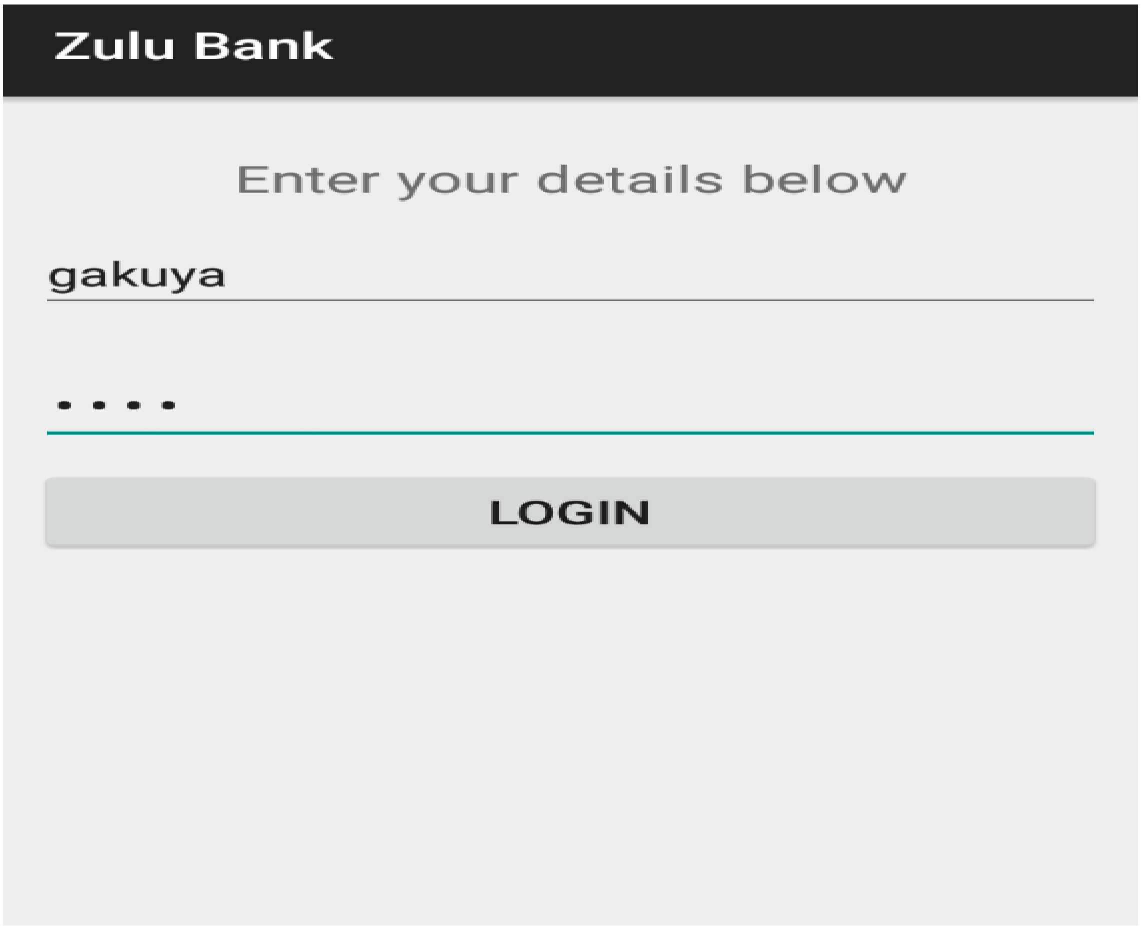
5.3 Prototype

Zulu bank is a mobile banking application prototype that provides secure mobile banking and payment services. It involves an administrator also referred to as the field officer creating different agents who cater for the needs and wants of customers.

The field officer has the authority of creating agents, suspending agents and reinstating them because of different occurrences. The field officer also oversees transactions carried by the customers through agents who deposit, withdraw, borrow loans and transfer funds upon providing their personal details, which includes the mobile number, identification card and their card numbers. Upon provision of these details and when the agent confirms the above credentials, a customer may then transact.

The mobile application also has a functionality that involves the use of a Near Field Communication (NFC) enabled member card, which points a customer's details to the banking application database. The agent needs to have a mobile phone, which is NFC enabled to use the NFC member cards.

Zulu bank provides security through authentication of the agent. In case the mobile phone may fall on the wrong hands, one is required to authenticate himself by inputting a username and a PIN to access the application services. At the application layer, before data leaves the device, the message is hashed and encrypted using the different hashing algorithms such as MD5 and SHA, and encryption algorithms like 3DES and AES. Security is therefore provided where data is not displayed in plaintext but in cipher text format. Figures 7, 8, and 9 show the different occurrences when the agent inputs his password on the mobile banking application.



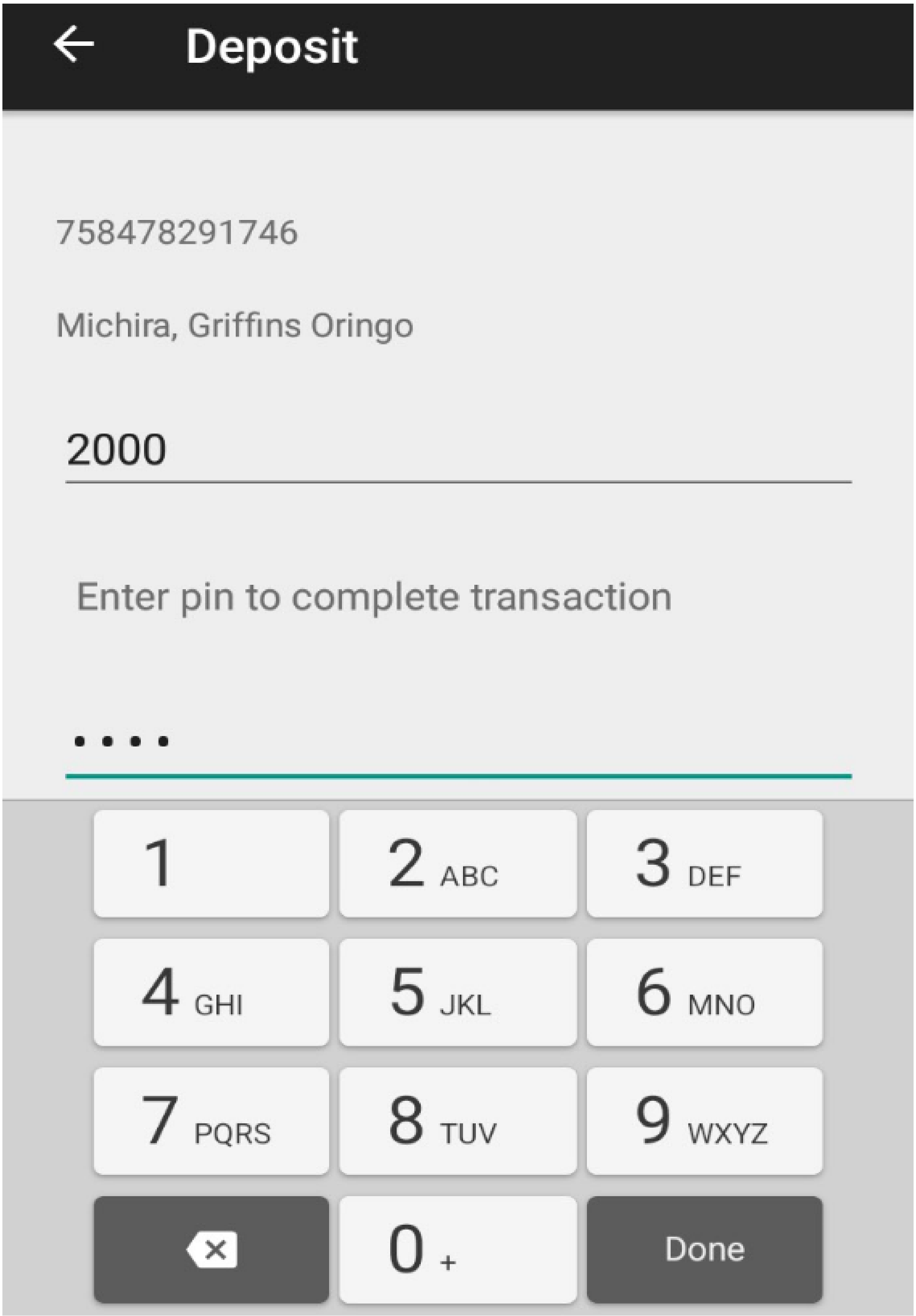
Zulu Bank

Enter your details below

gakuya

••••

Figure 6: PIN 1



Deposit

758478291746

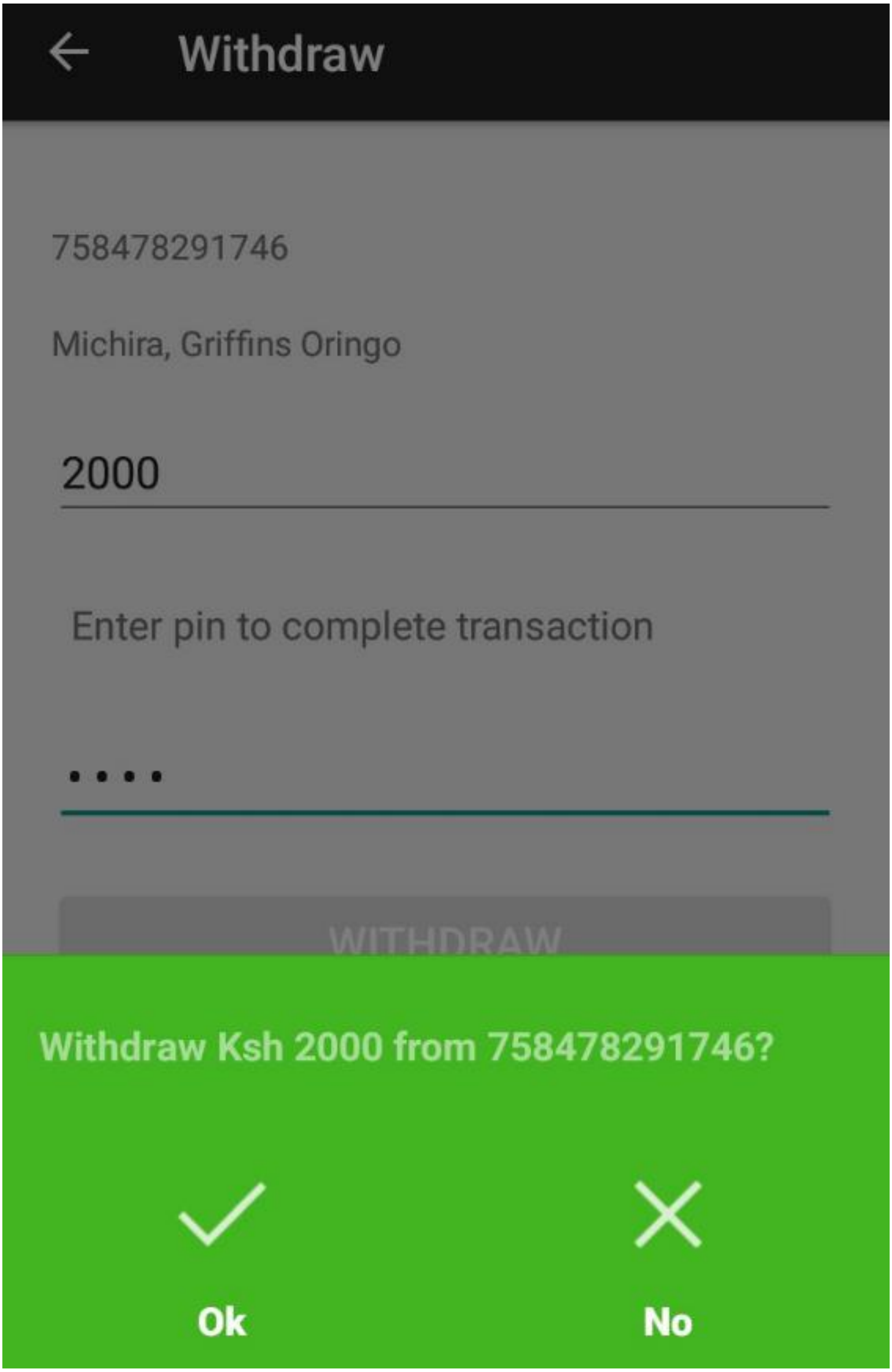
Michira, Griffins Oringo

2000

Enter pin to complete transaction

••••

Figure 7: PIN 2



758478291746

Michira, Griffins Oringo

2000

Enter pin to complete transaction

••••

Withdraw Ksh 2000 from 758478291746?

Figure 8: PIN 3

Software tokens manage sessions in Zulu bank. The sessions are managed when the agent logs on to the application and during transaction activities. Session timeouts are enabled in Zulu bank. When delays are experienced, the application would then timeout. This is an important feature in the application, whereby it reduces vulnerabilities to some of the known attacks such password cracking. From Figure 12 in the Appendix section 2, the use of software tokens is evident.

At the network layer, Zulu bank communicates using a secure mode of transmission that involves the use of the protocol Hyper Text Transfer Protocol Secure (HTTPS), which is HTTP on Secure Socket Layer (SSL). SSL uses digital certificates to verify the authenticity of the application server. Digital certificates provide secrecy and authentication. Digital signatures also ensure the integrity of messages that come from specific senders. SSL also provides confidentiality via encryption and authentication through the Message Authentication Code (MAC). Figure 14 on the Appendix Section 2 shows the uniform resource locator of Zulu bank as **https://www.zulubank.us.to** , which confirms the use of SSL.

Konheim (2007) says that it is very important to have a certificate, to authenticate a link in a network, especially a public network. SSL involves the use of keys when two devices are communicating. Keys should be sent via a reliable channel in ways the originator of the transmission would be known. Every user who wishes to receive private communication must place encryption algorithms (his public key) in the public file to facilitate confidentiality in communication and for the data being transported.

According to Konheim (2007), electronic transactions require digital signatures to be used on transactions. Electronic transactions especially in E-commerce are one of the reasons that led to the development and use of digital signatures and keys for authentication and provision of security at a larger scale.

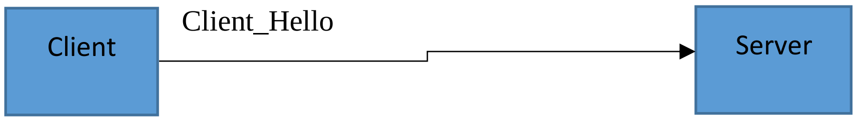
The use of keys was as a result of Merlkle and Hellman (1978) who provided the example of a public-key cryptosystem, and from most knowledge acquired this meets the needs and wants of a public key store system. Diffie and Hellman (1976) claimed that public key cryptography was invented to implement key exchange in a network, authentication of users and integrity of data during exchange of data.

Konheim (2007) maintains that the strength of most systems like RSA cryptosystems, depend on the difficulty of having large numbers, commonly prime number. The larger the prime number the safer it is to use the cryptosystem, in that it buys time in the event that one is trying to decipher a crypto system.

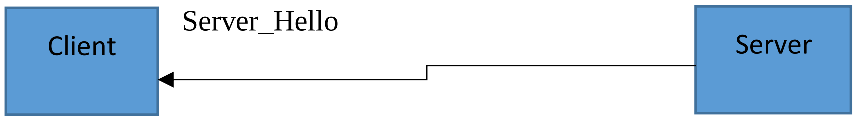
When a transaction is complete, Zulu bank would provide feedback in form of an SMS from the GSM network that would show the transaction details in terms of the amount of money transferred for the deposits, withdrawals, and funds transfer, loans credited or debited, the balance in the account, transaction number and time stamps. Both the agent and the account holder would get this message. The same details would be available on the webpage from the web and application server that used the Hypertext Transfer Protocol Secure (HTTPS) with the uniform resource locator as **https://www.zulubank.us.to**.

This was another security measure, which ensured the use of SSL for communication as discussed earlier between the mobile device and server. Data transferred between the mobile device and server was therefore encrypted. The below process shows how SSL communication between a client and server works during the process of exchanging data with the main aim of securing information to be communicated between a client and server.

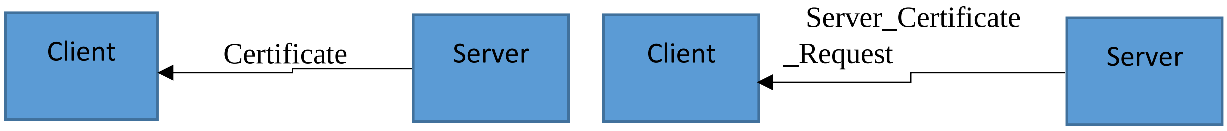
1. Key exchange protocols like RSA and Diffie-Hellman.
2. Data encryption algorithm like DES, 3DES, AES and RSA’s RC2 and RC4.
3. Message digest algorithms like MD5 and SHA.
4. A random number known as random bytes.
5. Session identifier.
6. Compression method identifier.



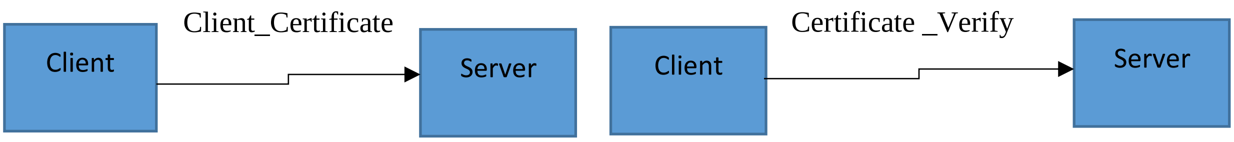
SSL Phase 1 (Client Hello).



Server response to Client_Hello.



SSL Phase 2 - server authentication and key exchange.



SSL phase 2 - client response to server.

Phase 2 – Client responds to the server. The Client delivers its digital certificate, which is a clear verification of the Server's certificate.

Zulu Bank utilized the use of hashing functions as explained earlier such as MD5, SHA1, SHA256 and SHA512 depending on the key, cipher and hash the mobile device and server support and preferred using during the communication process and exchange of data. The most secure hashing function, SHA512 supported by the two devices, the server and the client is selected as shown in Figure 9 in the Appendix 2 section.

Allan (2007, p. 470) states that a Message digest is a hash function that derives a fixed-length hash value for every message in some message domain. Hashing of data or a message protects the integrity of a message, in that if the hash of a message is different from what was the initial hash from a sender's message and what is obtained on the recipient's end then the message has been interfered with.

Zulu Bank utilized the use of hashing functions as explained earlier such as MD5, SHA1, SHA 256 and SHA512 depending on the key, cipher and hash the mobile device and server support and preferred using during the communication process and exchange of data. The most secure hashing function supported by the two devices, the server and client was selected. Figure 13 in the Appendix section 2 shows the use of a hash algorithm by Zulu Bank.

Secure Hashing Algorithm is commonly used in application protocols such as Transport Layer Security (TLS), Secure Socket Layer (SSL), Secure Shell (SSH) and Internet Protocol Security. Hashing provides security by having one-way hash function property, in that when data is hashed there is no way of reversing it. Secondly, one cannot hash a message for the second time and get the same hash result. The above hashing properties are contradicted by the rainbow table attack, which has been known to produce plaintext information from data that had been initially hashed such as the PIN of a user.

Hashing makes data or messages in transit or at rest secure in that it cannot be read in clear text in case it falls on the wrong hands. Examples of hashing algorithms as discussed earlier are Message Digest 5 (MD5) by RSA and Secure Hashing Algorithm (SHA) by NIST. Devices agree on how to encrypt or hash data as shown on the table below based on the algorithm that is most secure and is supported by both parties, in this case the mobile application and the application server.

Key	**Cipher**	**Hash**
RSA	RC4	HMAC- MD5
Diffie-Hellman	3 DES	HMAC- SHA
DSA	AES	

Table 13: Algorithms

CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusion

This study demonstrated the insecurity of mobile banking and payments in Kenya by performing penetration testing using Charles Proxy and Mobile Application Reverse Engineering and Analysis Framework tools. The penetration testing was done on six mobile banking and payment applications for individual banks in the Tier 1 category.

One hundred percent of the financial institutions in the Tier 1 category were found to have embraced mobile banking and payment solutions. It is evident that financial institutions are trying to cope with the dynamic nature of technology. This has resulted in to provision of convenient services to customers at the expense of security. A cause of disagreement exists in providing convenient and secure services or secure and convenient services.

From the penetration testing done, it was evident that most of the M-banking applications had not achieved the Open Web Application Security Project standards during development and before the release of these applications to the market. Solutions were provided for these vulnerabilities, which involved the development of a secure model and application prototype. The model and prototype demonstrated how security in both the application and network layers could be achieved for data protection.

It would be advisable for an application to achieve **100%** of security standards such as OWASP before release of the application to the market, having in mind that it is almost impossible to have a fully secure mobile application, especially for applications that use the network for connectivity and communication purposes. This is because threats and vulnerabilities of applications are created and discovered after short periods. This research attests that Confidentiality, Integrity and Availability are the major pillars of security. To achieve security, challenges with the CIA triad have to be addressed.

6.2 Limitations of the project

Penetration testing was only done on M-banking applications, which use the android operating system. Devices and applications that support other operating system such as Windows, iOS and Blackberry were not part of the research. This was a limitation because other mobile banking applications use these operating systems and we were not able to get information on these applications and their operating systems so they were not part of this study.

In this research, the application prototype, did not address issues to do with rainbow table attacks. This is an attack that is used to compromise the hash of a password. The remedy for this attack would be salting of a password, which involves a username being connected to a password, then appending a random value of data to the password, which is then hashed. This would be a more secure method for password protection, whereby the random value added provides more security instead of just hashing the password that was initially in plaintext.

6.3 Recommendations

Mobile banking and payment applications are to achieve the Open Web Application Security Project (OWASP) standards during the process of their development and before their release. It is recommended to develop applications and systems that support multiple authentication schemes to provide security in both the application and network layers.

It is also recommended that financial institutions, through their Information Technology experts especially information security experts, should pay full attention to matters of security by analysing financial applications and systems to be able to deliver convenient and secure services to their customers. This can be achieved by addressing the issues of privacy and confidentiality by conducting security audits frequently.

End users who use M-banking and payment solutions, have to be educated on the different forms of cyber-attacks such as social engineering and how they can secure themselves from these attacks. Financial institutions through their communication channels can play this role to secure their customers.

Further research is also recommended on ways of enhancing security of mobile banking and payments by providing end-to-end security, to strike a balance in offering secure and convenient services.

References

1. Aslam, H.M., 2009. Mobile payments: Will Colombo keep its leadership in South Asia.

2. Chikomo, K., Chong, M.K., Arnab, A. and Hutchison, A., 2006. Security of mobile banking.*University of Cape Town, South Africa, Tech. Rep., Nov, 1*.

3. Chong, M.K., 2006. Security of mobile banking: Secure SMS banking. *Data Network Architectures Group. University of Cape Town, South Africa*.

4. Dahlberg, T., Mallat, N., Ondrus, J. and Zmijewska, A., 2008. Past, present and future of mobile payments research: A literature review. *Electronic Commerce Research and Applications*, *7*(2), pp.165-181.
5. Emmanuel, A. and Jacobs, B., 2007. Mobile Banking in Developing Countries: Secure Framework for Delivery of SMS-banking Services. *Radboud University Nijmegan, The Netherland*.

6. Gódor, G., Faigl, Z. and Szalay, M., 2009. Mobile payment. *Encyclopedia of Information Science and Technology*.
7. Goulder, M.H., 2011. Network Defense: Security and Vulnerability Assessment. Course Technology.

8. Kahn, D., 1996. *The Codebreakers: The Story of Secret Writing from Ancient Times to the Internet*. Scribner.

9. Kamal, R., 2012, Mobile Computing, 2ndedn., Oxford University Press, India.

10. Kaur, G., Kaur, P. and Saluja, K.K., 2012. A Review of Security issues and mitigation Measures in GSM. *International Journal of Research in Engineering & Applied Sciences*, *2*(2), p.16.

11. Kaur, G., Sachdeva, M. and Saluja, K.K., 2012. Mobile Communication: SMS Security Review (Issues, Attacks and Preventive Measures). *Networking and Communication Engineering*.

12. Konheim, A.G., 2007. *Computer security and cryptography*. John Wiley & Sons.

13. Krugel, G.T., 2007. Mobile Banking Technology Options.*FinMark Trust*.

14. Kumar, S., Ramesh, B. and Rabara, S.A., 2010. An architectural design for secure mobile remote macro-payments. *Journal of Next Generation Information Technology*, *1*(2).

15. Kumar, S.B.R., Raj, A. and Rabara, S.A., 2008, December. A framework for mobile payment consortia system (MPCS). In *Computer Science and Software Engineering, 2008 International Conference on* (Vol. 2, pp. 43-47). IEEE.

16. Langer, J. and Roland, M., 2010. Architektur mobiler NFC-Geräte. *Anwendungen und Technik von Near Field Communication (NFC).*

17. Lehtinen, R. and Gangemi Sr, G.T., 2006. *Computer security basics*. " O'Reilly Media, Inc.".

18. Mbengo, P. and Phiri, M.A., 2015. MOBILE BANKING ADOPTION: A RURAL ZIMBABWEAN MARKETING PERSPECTIVE.

19. MEUCCI, M. and Muller, A., 2014. The OWASP Testing Guide 4.0. *no. Cc*.

20. MSc–imran, I.A., RE, R.M. and CISA, E.R.R., Mobile Banking Security.

21. Narendiran, C., Rabara, S.A. and Rajendran, N., 2009, October. Public key infrastructure for mobile banking security. In *Global Mobile Congress 2009*(pp. 1-6). IEEE.

22. Nguyen, N., 2014. Potential of developing and using mobile banking apps in Vietnam.

23. Nicoletti, B., 2014. *Mobile Banking: Evolution or Revolution?* Springer.

24. Raemaenen, J., 2011. *Perceived security in mobile authentication* (Doctoral dissertation, Master’s thesis, Aalto University, School of Electrical Engeineering).

25. Stamp, M., 2011. *Information security: principles and practice*. John Wiley & Sons.
26. Van der Merwe, P.B., 2005. Mobile commerce over GSM: A banking perspective on security.

27. Zhang, F., 2012. Secure Mobile Service-Oriented Architecture

APPENDIX 1

```
*****************************************************************************
**   AndroBugs Framework - Android App Security Vulnerability Scanner  **dfx
**                          version: 1.0.0                      **
**    author: Yu-Cheng Lin (@AndroBugs, http://www.AndroBugs.com)     **
**              contact: androbugs.framework@gmail.com              **
*****************************************************************************
Platform: Android
Package Name:
Package Version Name: 3.0.8
Package Version Code: 58
Min Sdk: 14
Target Sdk: 22
MD5: cfc90ff05bbc2b48a52b6f7fee2e6706
SHA1: 07262a512917c889319d6e044d430986115ff534
SHA256: 0f677b7d56c0ffc19678d96d9b14cbb98f69fe0035505870f3dabbc11db93637
SHA512:
21a07a5320dd6a871b1c94ee8f01a84fcbc61c10a78187f81b53dd5a8ff5dd345ac1b07bf40c4e0
f2a6b6594f9ef3689dc486980198966bed786f8e562797544
Analyze Signature:
d70dbb1f4d7ec775c496fbcf0ce30cd271dffd777a50a4039bcb157a2ae3b8af6319a31868d5fcb
1f1cada98a30eaefe9f90a3c9a781d354ddac6e10baf554be
------------------------------------------------------------------------------------------
[Critical] <KeyStore><Hacker> KeyStore Protection Checking:
The Keystores below seem using "byte array" or "hard-coded cert info" to do SSL pinning
(Total: 1). Please manually check:
=> Lcom/kcbbankgroup/android/ard;->a(Landroid/content/Context;
Ljava/lang/String;)Ljavax/net/ssl/HttpsURLConnection; (0x6a)
---> Ljava/security/KeyStore;->load(Ljava/io/InputStream; [C)V
[Critical] <Implicit_Intent> Implicit Service Checking:
To ensure your app is secure, always use an explicit intent when starting a Service and DO
NOT declare intent filters for your
```

services. Using an implicit intent to start a service is a security hazard because you cannot be certain what service will

respond to the intent, and the user cannot see which service starts.

Reference: http://developer.android.com/guide/components/intents-filters.html#Types

=> com.abello.fcm.MyFirebaseMessagingService

=> com.abello.fcm.MyFirebaseInstanceIDService

=> com.google.firebase.iid.FirebaseInstanceIdService

=> com.google.firebase.messaging.FirebaseMessagingService

[Critical] AndroidManifest ContentProvider Exported Checking:

We strongly suggest you explicitly specify the "exported" attribute (AndroidManifest.xml).

For Android "android:targetSdkVersion" < 17, the exported value of ContentProvider is "true" by default.

For Android "android:targetSdkVersion" >= 17, the exported value of ContentProvider is "false" by default.

Which means if you do not explicitly set the "android:exported", you will expose your ContentProvider to Android < 4.2 devices.

Even if you set the provider the permission with [protectionalLevel="normal"], other apps still cannot access it on Android >=

4.2 devices because of the default constraint.

Please make sure to set exported to "true" if you initially want other apps to use it (including protected by "signature"

protectionalLevel), and set to "false" if your do not want to.

Please still specify the "exported" to "true" if you have already set the corresponding "permission", "writePermission" or

"readPermission" to "signature" protectionLevel or higher

because other apps signed by the same signature in Android >= 4.2 devices cannot access it.

Reference: http://developer.android.com/guide/topics/manifest/provider-element.html#exported

Vulnerable ContentProvider Case Example:

(1)https://www.nowsecure.com/mobile-security/ebay-android-content-provider-injection-vulnerability.html

(2)http://blog.trustlook.com/2013/10/23/ebay-android-content-provider-information-disclosure-vulnerability/

(3)http://www.wooyun.org/bugs/wooyun-2010-039169

```
[Critical] <SSL_Security> SSL Connection Checking:
http://play.google.com/store/apps/details
=> Lcom/google/android/youtube/player/a/bf;-><clinit>()V
http://plus.google.com/
=> Lcom/google/android/gms/common/internal/ba;-><clinit>()V
http://www.youtube.com/watch?v=(Most of these websites are third party).
=> Lcom/kocela/android/bank/video/a;->a(Lcom/kocela/android/bank/video/d;)V
=> Lcom/kocela/android/bank/video/YoutubePlayerViewActivity;->b()V
[Warning]  External Storage Accessing:
External storage access found (Remember DO NOT write important files to external storages):
; Ljava/util/ArrayList;)Ljava/io/File; (0xac) --->
Landroid/os/Environment;->getExternalStorageDirectory()Ljava/io/File;
[Warning]  AndroidManifest Exported Components Checking:
Found "exported" components(except for Launcher) for receiving outside applications' actions (AndroidManifest.xml).
These components can be initilized by other apps. You should add or modify the attribute to [exported="false"] if you don't want
to.
You can also protect it with a customized permission with "signature" or higher protectionLevel and specify in
"android:permission" attribute.
activity => com.google.android.gms.appinvite.PreviewActivity
service => com.abello.fcm.MyFirebaseMessagingService
service => com.abello.fcm.MyFirebaseInstanceIDService
service => com.google.firebase.iid.FirebaseInstanceIdService
service => com.google.firebase.messaging.FirebaseMessagingService
receiver => com.google.android.gms.measurement.AppMeasurementReceiver
[Warning] <Sensitive_Information> Getting ANDROID_ID:
This app has code getting the 64-bit number "Settings.Secure.ANDROID_ID".
ANDROID_ID seems a good choice for a unique device identifier. There are downsides: First, it is not 100% reliable on releases of
Android prior to 2.2 (Froyo).
Also, there has been at least one widely-observed bug in a popular handset from a major manufacturer, where every instance has
```

the same ANDROID_ID.

If you want to get an unique id for the device, we suggest you use "Installation" framework in the following article.

Please check the reference: http://android-developers.blogspot.tw/2011/03/identifying-app-installations.html

=> Lcom/google/android/gms/measurement/internal/ce;->a(Lcom/google/android/gms/measurement/internal/aw;

Lcom/google/android/gms/measurement/internal/AppMetadata;)V (0x312) --->

Landroid/provider/Settings$Secure;->getString(Landroid/content/ContentResolver;

Ljava/lang/String;)Ljava/lang/String;

[Warning] <WebView> WebView Local File Access Attacks Checking:

Found "setAllowFileAccess(true)" or not set(enabled by default) in WebView. The attackers could inject malicious script into

WebView and exploit the opportunity to access local resources.This can be mitigated or prevented by disabling local file system

access. (It is enabled by default)

Note that this enables or disables file system access only. Assets and resources are still accessible using file:///android_asset

and file:///android_res.

The attackers can use "mWebView.loadUrl("file:///data/data/[Your_Package_Name]/[File]");" to access app's local file.

Reference: (1)https://labs.mwrinfosecurity.com/blog/2012/04/23/adventures-with-android-webviews/

(2)http://developer.android.com/reference/android/webkit/WebSettings.html#setAllowFileAccess(boolean)

Please add or modify "yourWebView.getSettings().setAllowFileAccess(false)" to your WebView:

[Warning] <WebView> WebView Potential XSS Attacks Checking:

Found "setJavaScriptEnabled(true)" in WebView, which could exposed to potential XSS attacks. Please check the web page code

carefully and sanitize the output:

(0x134) --->

Landroid/webkit/WebSettings;->setJavaScriptEnabled(Z)V

Landroid/webkit/WebSettings;->setJavaScriptEnabled(Z)V

[Notice] AndroidManifest Adb Backup Checking:

ADB Backup is ENABLED for this app (default: ENABLED). ADB Backup is a good tool for backing up all of your files. If it's open

for this app, people who have your phone can copy all of the sensitive data for this app in your phone(Prerequisite: 1.Unlock

phone's screen 2.Open the developer mode). The sensitive data may include lifetime access token, username or password, etc.

Security case related to ADB Backup:

1.

2.http://blog.c22.cc/advisories/cve-2013-5112-evernote-android-insecure-storage-of-pin-data-bypass-of-pin-protection/

3.http://nelenkov.blogspot.co.uk/2012/06/unpacking-android-backups.html

Reference: http://developer.android.com/guide/topics/manifest/application-element.html#allowbackup

[Notice] <Database><#CVE-2011-3901#> Android SQLite Databases Vulnerability Checking:

This app is using Android SQLite databases.

Prior to Android 4.0, Android has SQLite Journal Information Disclosure Vulnerability.

But it can only be solved by users upgrading to Android > 4.0 and YOU CANNOT SOLVE IT BY YOURSELF (But you can use encrypt your

databases and Journals by "SQLCipher" or other libs).

Proof-Of-Concept Reference:

(1) http://blog.watchfire.com/files/androidsqlitejournal.pdf

(2) http://www.youtube.com/watch?v=oCXLHjmH5rY

[Notice] File Unsafe Delete Checking:

Everything you delete may be recovered by any user or attacker, especially rooted devices.

Please make sure do not use "file.delete()" to delete essential files.

Check this video: https://www.youtube.com/watch?v=tGw1fxUD-uY

=> Lcom/google/android/gms/analytics/internal/ar;->getWritableDatabase()Landroid/database/sqlite/SQLiteDatabase; (0x7a) --->

Ljava/io/File;->delete()Z

=> Lcom/google/android/gms/measurement/internal/as;->getWritableDatabase()Landroid/database/sqlite/SQLiteDatabase; (0x98)

```
---> Ljava/io/File;->delete()Z
[Notice] <Hacker> APK Installing Source Checking:
This app has code checking APK installer sources(e.g. from Google Play, from Amazon, etc.).
It might be used to check for whether
the app is hacked by the attackers.
=> Lcom/google/android/gms/analytics/ac;->a()Lcom/google/android/gms/c/n; (0x3a) --->
Landroid/content/pm/PackageManager;-
>getInstallerPackageName(Ljava/lang/String;)Ljava/lang/String;
=> Lcom/google/android/gms/measurement/internal/bf;->e()V (0x30) --->
Landroid/content/pm/PackageManager;-
>getInstallerPackageName(Ljava/lang/String;)Ljava/lang/String;
[Notice] <Signature><Hacker> Getting Signature Code Checking:
This app has code checking the package signature in the code. It might be used to check for
whether the app is hacked by the
attackers.
=> Lcom/google/android/gms/common/a/n;->a(Landroid/content/Context; I)Z (0x24) --->
Landroid/content/pm/PackageManager;->getPackageInfo(Ljava/lang/String;
I)Landroid/content/pm/PackageInfo;
=> Lcom/google/android/gms/common/r;->c(Landroid/content/Context;)I (0x48) --->
Landroid/content/pm/PackageManager;->getPackageInfo(Ljava/lang/String;
I)Landroid/content/pm/PackageInfo;
=> Lcom/google/android/gms/measurement/internal/bf;->E()Z (0x24) --->
Landroid/content/pm/PackageManager;->getPackageInfo(Ljava/lang/String;
I)Landroid/content/pm/PackageInfo;
=> Lcom/google/android/gms/measurement/internal/bf;->e()V (0x216) --->
Landroid/content/pm/PackageManager;->getPackageInfo(Ljava/lang/String;
I)Landroid/content/pm/PackageInfo;
=>                                                    Lcom/google/android/youtube/player/a;-
>a(Landroid/content/Context;)Lcom/google/android/youtube/player/b; (0x14) --->
Landroid/content/pm/PackageManager;->getPackageInfo(Ljava/lang/String;
I)Landroid/content/pm/PackageInfo;
[Notice]  AndroidManifest Exported Components Checking 2:
Found "exported" components(except for Launcher) for receiving Google's "Android" actions
(AndroidManifest.xml):
```

[Info] <Command> Runtime Command Checking:

This app is not using critical function 'Runtime.getRuntime().exec("...")'.

[Info] <Command> Executing "root" or System Privilege Checking:

Did not find codes checking "root" permission(su) or getting system permission (It's still possible we did not find out).

[Info] <Database> SQLiteDatabase Transaction Deprecated Checking:

Ignore checking "SQLiteDatabase:beginTransactionNonExclusive" because your set minSdk >= 11.

[Info] <Database> Android SQLite Databases Encryption (SQLite Encryption Extension (SEE)):

This app is "NOT" using SQLite Encryption Extension (SEE) on Android (http://www.sqlite.org/android) to encrypt or decrpyt

databases.

[Info] <Database> Android SQLite Databases Encryption (SQLCipher):

This app is "NOT" using SQLCipher(http://sqlcipher.net/) to encrypt or decrpyt databases.

[Info] <Debug> Android Debug Mode Checking:

DEBUG mode is OFF(android:debuggable="false") in AndroidManifest.xml.

[Info] Dynamic Code Loading:

No dynamic code loading(DexClassLoader) found.

[Info] <#BID 64208, CVE-2013-6271#> Fragment Vulnerability Checking:

Did not detect the vulnerability of "Fragment" dynamically loading into "PreferenceActivity" or "SherlockPreferenceActivity"

[Info] <Framework> Framework - MonoDroid:

This app is NOT using MonoDroid Framework (http://xamarin.com/android).

[Info] <Hacker> Base64 String Encryption:

No encoded Base64 String or Urls found.

[Info] <Database><Hacker>Key for Android SQLite Databases Encryption:

Did not find using the symmetric key(PRAGMA key) to encrypt the SQLite databases (It's still possible that it might use but we

did not find out).

[Info] <Debug><Hacker> Codes for Checking Android Debug Mode:

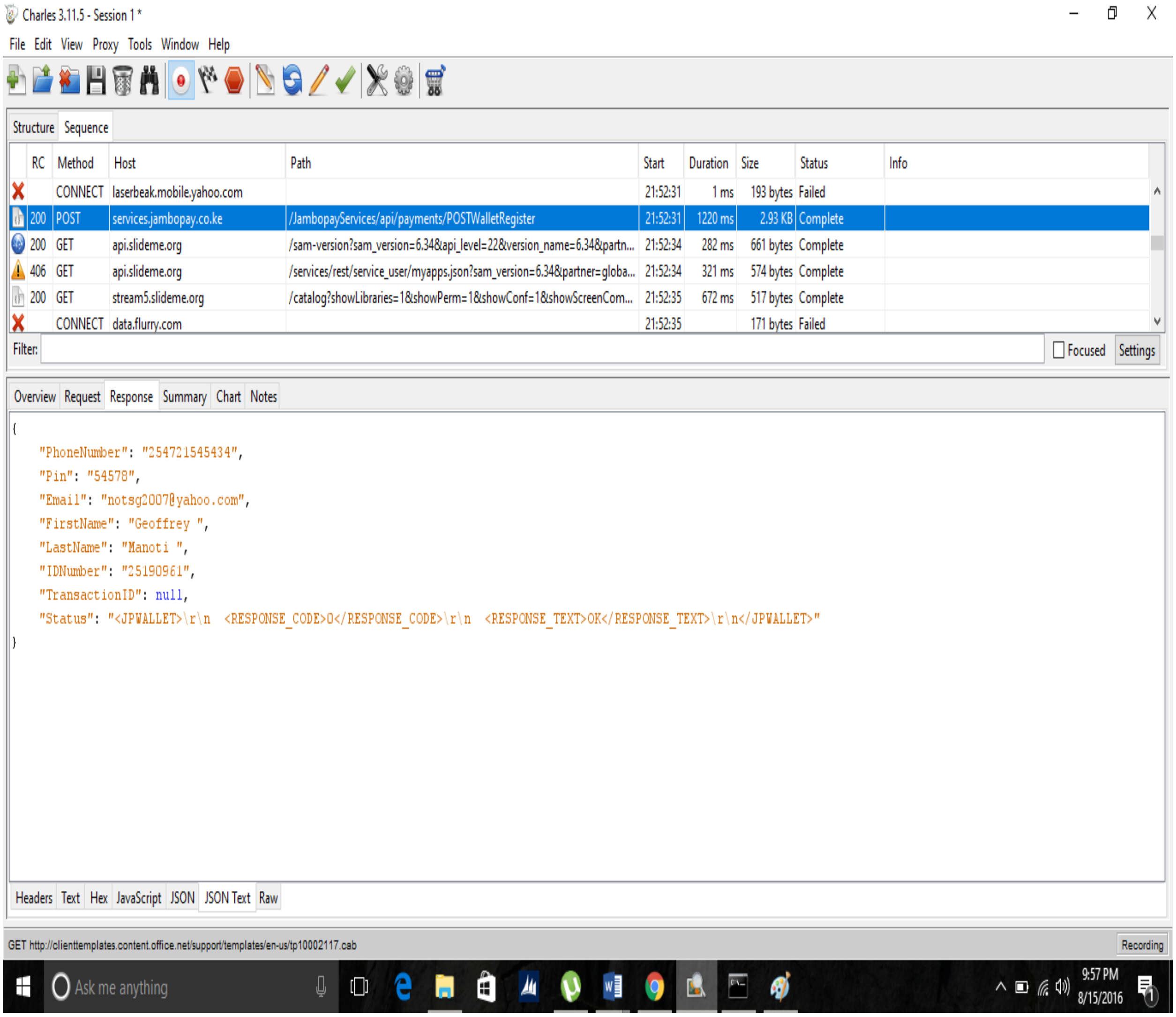
Did not detect codes for checking "ApplicationInfo.FLAG_DEBUGGABLE" in AndroidManifest.xml.

```
[Info] <KeyStore><Hacker> KeyStore File Location:
Did not find any possible BKS keystores or certificate keystore file (Notice: It does not mean
this app does not use keysotre):
[Info] <Hacker>Code Setting Preventing Screenshot Capturing:
Did not detect this app has code setting preventing screenshot capturing.
[Info]  HttpURLConnection Android Bug Checking:
Ignore checking "http.keepAlive" because you're not using "HttpURLConnection" and
min_Sdk > 8.
[Info] <KeyStore> KeyStore Type Checking:
KeyStore 'BKS' type check OK
[Info]  Google Cloud Messaging Suggestion:
Nothing to suggest.
[Info] <#CVE-2013-4787#> Master Key Type I Vulnerability:
No Master Key Type I Vulnerability in this APK.
[Info]  App Sandbox Permission Checking:
No security issues "MODE_WORLD_READABLE" or "MODE_WORLD_WRITEABLE"
found on 'openOrCreateDatabase' or 'openOrCreateDatabase2' or
'getDir' or 'getSharedPreferences' or 'openFileOutput'
[Info]  Native Library Loading Checking:
No native library loaded.
[Info]  AndroidManifest Dangerous ProtectionLevel of Permission Checking:
No "dangerous" protection level customized permission found (AndroidManifest.xml).
[Info]  AndroidManifest PermissionGroup Checking:
PermissionGroup in permission tag of AndroidManifest sets correctly.
[Info]  AndroidManifest "intent-filter" Settings Checking:
"intent-filter" of AndroidManifest.xml check OK.
[Info]  AndroidManifest Normal ProtectionLevel of Permission Checking:
No default or "normal" protection level customized permission found (AndroidManifest.xml).
[Info] <#CVE-2013-6272#> AndroidManifest Exported Lost Prefix Checking:
No exported components that forgot to add "android:" prefix.
[Info] <Sensitive_Information> Getting IMEI and Device ID:
Did not detect this app is getting the "device id(IMEI)" by "TelephonyManager.getDeviceId()"
approach.
[Info]  Codes for Sending SMS:
```

```
Did not detect this app has code for sending SMS messages (sendDataMessage, sendMultipartTextMessage or sendTextMessage).
[Info] <System> AndroidManifest sharedUserId Checking:
This app does not use "android.uid.system" sharedUserId.
[Info] <SSL_Security> SSL Implementation Checking (Verifying Host Name in Custom Classes):
Self-defined HOSTNAME VERIFIER checking OK.
[Info] <SSL_Security> SSL Implementation Checking (Verifying Host Name in Fields):
Critical vulnerability "ALLOW_ALL_HOSTNAME_VERIFIER" field setting or "AllowAllHostnameVerifier" class instance not found.
[Info] <SSL_Security> SSL Implementation Checking (Insecure component):
Did not detect SSLSocketFactory by insecure method "getInsecure".
[Info] <SSL_Security> SSL Implementation Checking (HttpHost):
DEFAULT_SCHEME_NAME for HttpHost check: OK
[Info] <SSL_Security> SSL Implementation Checking (WebViewClient for WebView):
Did not detect critical usage of "WebViewClient"(MITM Vulnerability).
[Info] <SSL_Security>SSL Certificate Verification Checking:
Did not find vulnerable X509Certificate code.
[Info]  Unnecessary Permission Checking:
Permission 'android.permission.ACCESS_MOCK_LOCATION' sets correctly.
[Info]  Accessing the Internet Checking:
This app is using the Internet via HTTP protocol.
[Info]  AndroidManifest System Use Permission Checking:
No system-level critical use-permission found.
[Info]  <WebView><Remote Code Execution><#CVE-2013-4710#> WebView RCE Vulnerability Checking:
WebView addJavascriptInterface vulnerabilities not found.
------------------------------------------------------------
AndroBugs a
nalyzing time: 10.24725 secs
Total elapsed time: 40.258168 secs
```

N.B The sections highlighted in red are the critical warning flags, in yellow are the mild warning flags and in purple are the positive characteristics of the application.

APPENDIX 2


Charles 3.11.5 - Session 1 *
File Edit View Proxy Tools Window Help
Structure Sequence
Overview Request Response Summary Chart Notes
Headers Text Hex JavaScript JSON JSON Text Raw
Recording
Ask me anything
9:57 PM
8/15/2016

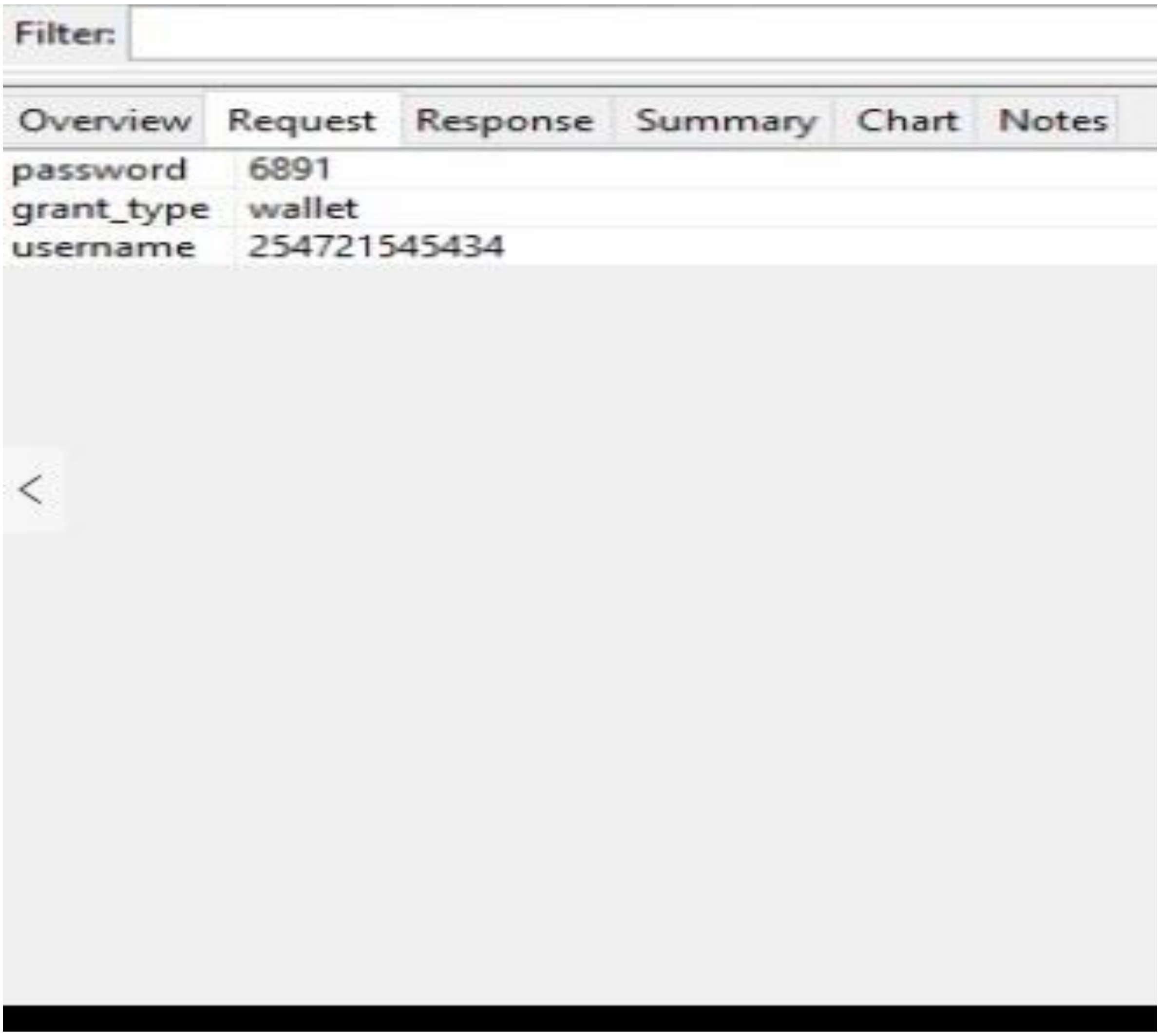
	RC	Method	Host	Path	Start	Duration	Size	Status	Info
✗		CONNECT	laserbeak.mobile.yahoo.com		21:52:31	1 ms	193 bytes	Failed	
	200	POST	services.jambopay.co.ke	/JambopayServices/api/payments/POSTWalletRegister	21:52:31	1220 ms	2.93 KB	Complete	
	200	GET	api.slideme.org	/sam-version?sam_version=6.34&api_level=22&version_name=6.34&partn...	21:52:34	282 ms	661 bytes	Complete	
⚠	406	GET	api.slideme.org	/services/rest/service_user/myapps.json?sam_version=6.34&partner=globa...	21:52:34	321 ms	574 bytes	Complete	
	200	GET	stream5.slideme.org	/catalog?showLibraries=1&showPerm=1&showConf=1&showScreenCom...	21:52:35	672 ms	517 bytes	Complete	
✗		CONNECT	data.flurry.com		21:52:35		171 bytes	Failed	

Filter: □ Focused Settings

```
{
    "PhoneNumber": "254721545434",
    "Pin": "54578",
    "Email": "notsg2007@yahoo.com",
    "FirstName": "Geoffrey ",
    "LastName": "Manoti ",
    "IDNumber": "25190961",
    "TransactionID": null,
    "Status": "<JPWALLET>\r\n  <RESPONSE_CODE>0</RESPONSE_CODE>\r\n  <RESPONSE_TEXT>OK</RESPONSE_TEXT>\r\n</JPWALLET>"
}
```

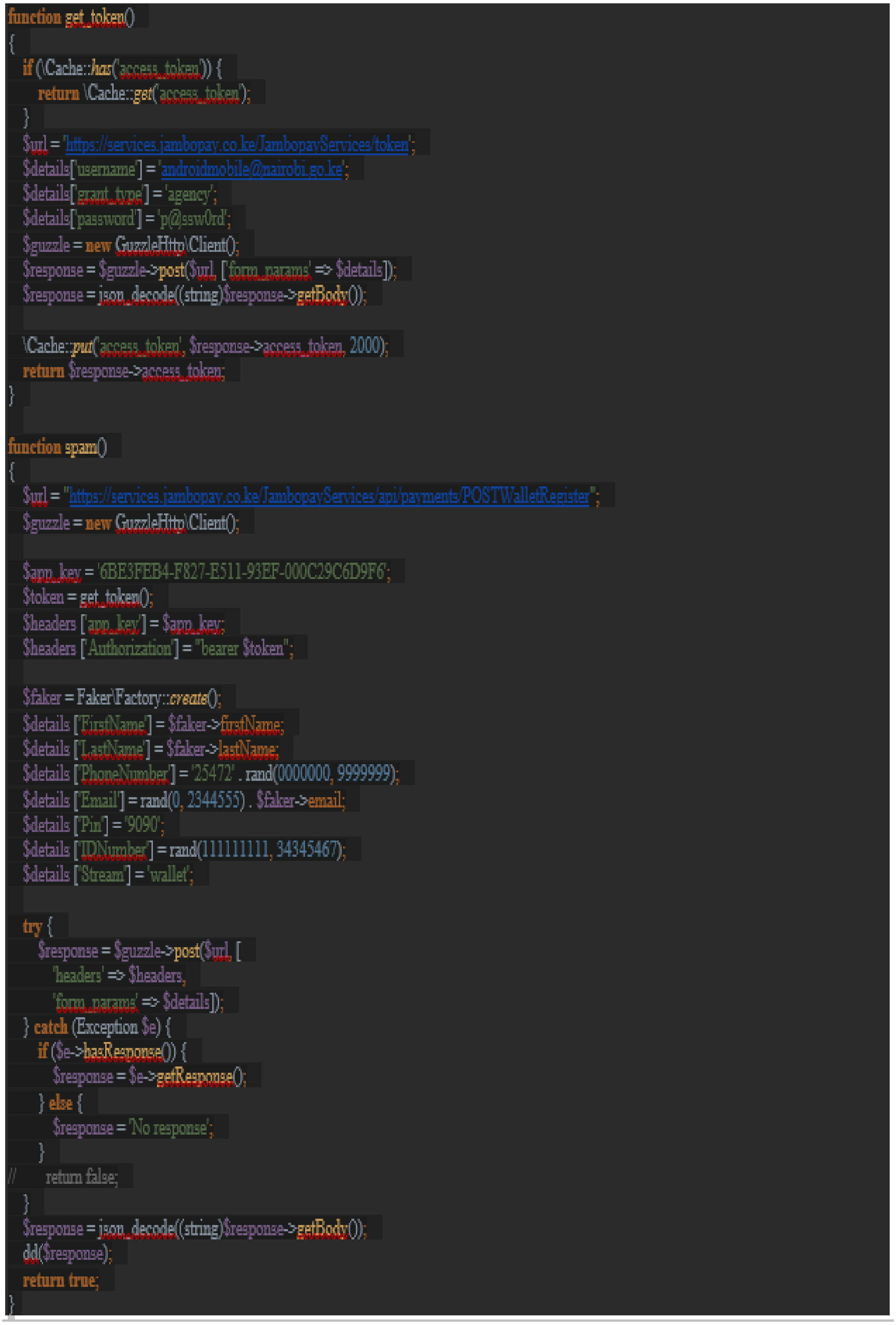
GET http://clienttemplates.content.office.net/support/templates/en-us/tp10002117.cab

Figure 9: Vulnerable App1



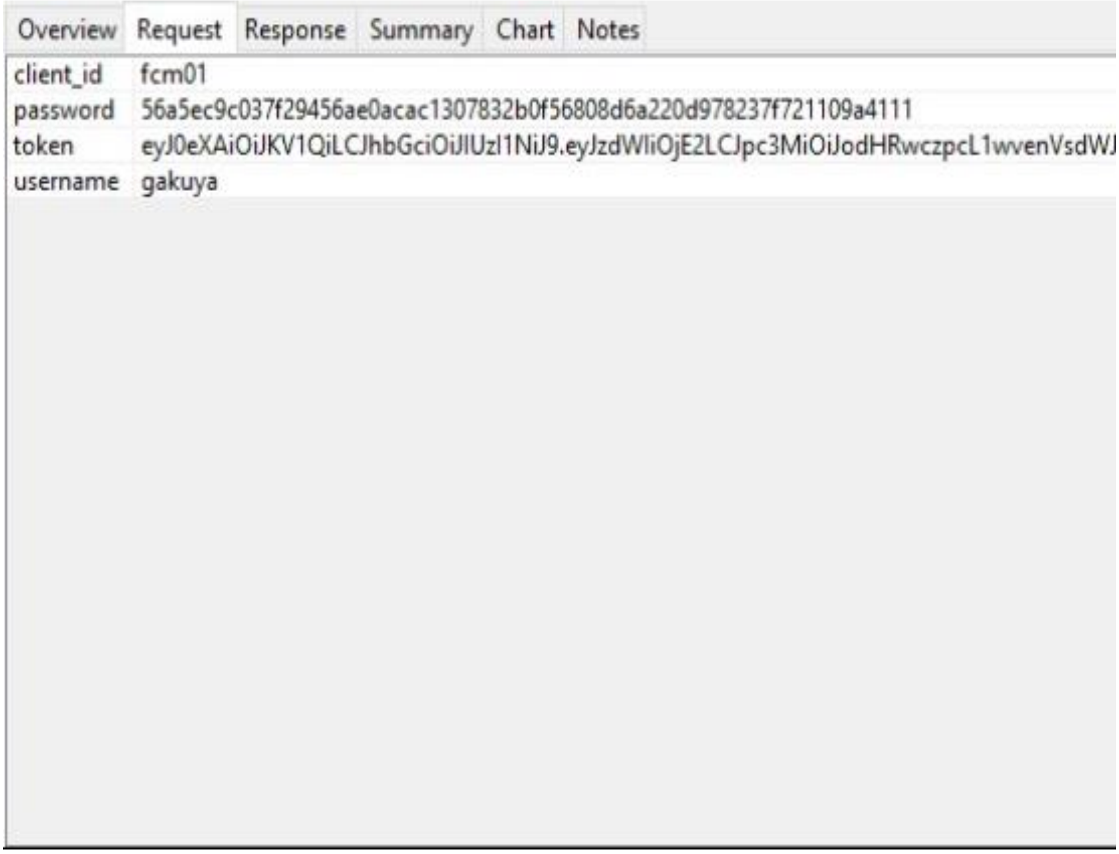
password	6891
grant_type	wallet
username	254721545434

Figure 10: Vulnerable App2



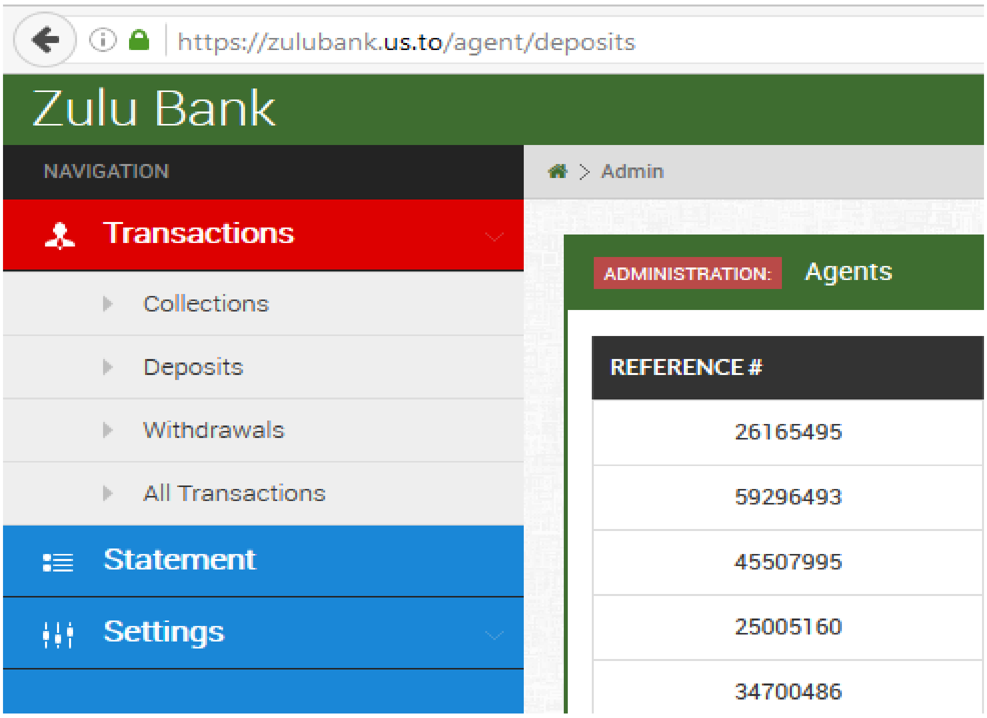
```php
function get_token()
{
   if (\Cache::has('access_token')) {
      return \Cache::get('access_token');
   }
   $url = 'https://services.jambopay.co.ke/JambopayServices/token';
   $details['username'] = 'androidmobile@nairobi.go.ke';
   $details['grant_type'] = 'agency';
   $details['password'] = 'p@ssw0rd';
   $guzzle = new GuzzleHttp\Client();
   $response = $guzzle->post($url, ['form_params' => $details]);
   $response = json_decode((string)$response->getBody());

   \Cache::put('access_token', $response->access_token, 2000);
   return $response->access_token;
}

function spam()
{
   $url = "https://services.jambopay.co.ke/JambopayServices/api/payments/POSTWalletRegister";
   $guzzle = new GuzzleHttp\Client();

   $app_key = '6BE3FEB4-F827-E511-93EF-000C29C6D9F6';
   $token = get_token();
   $headers ['app_key'] = $app_key;
   $headers ['Authorization'] = "bearer $token";

   $faker = Faker\Factory::create();
   $details ['FirstName'] = $faker->firstName;
   $details ['LastName'] = $faker->lastName;
   $details ['PhoneNumber'] = '25472' . rand(0000000, 9999999);
   $details ['Email'] = rand(0, 2344555) . $faker->email;
   $details ['Pin'] = '9090';
   $details ['IDNumber'] = rand(111111111, 34345467);
   $details ['Stream'] = 'wallet';

   try {
      $response = $guzzle->post($url, [
         'headers' => $headers,
         'form_params' => $details]);
   } catch (Exception $e) {
      if ($e->hasResponse()) {
         $response = $e->getResponse();
      } else {
         $response = 'No response';
      }
//       return false;
   }
   $response = json_decode((string)$response->getBody());
   dd($response);
   return true;
}
```

Figure 11: Vulnerable App3



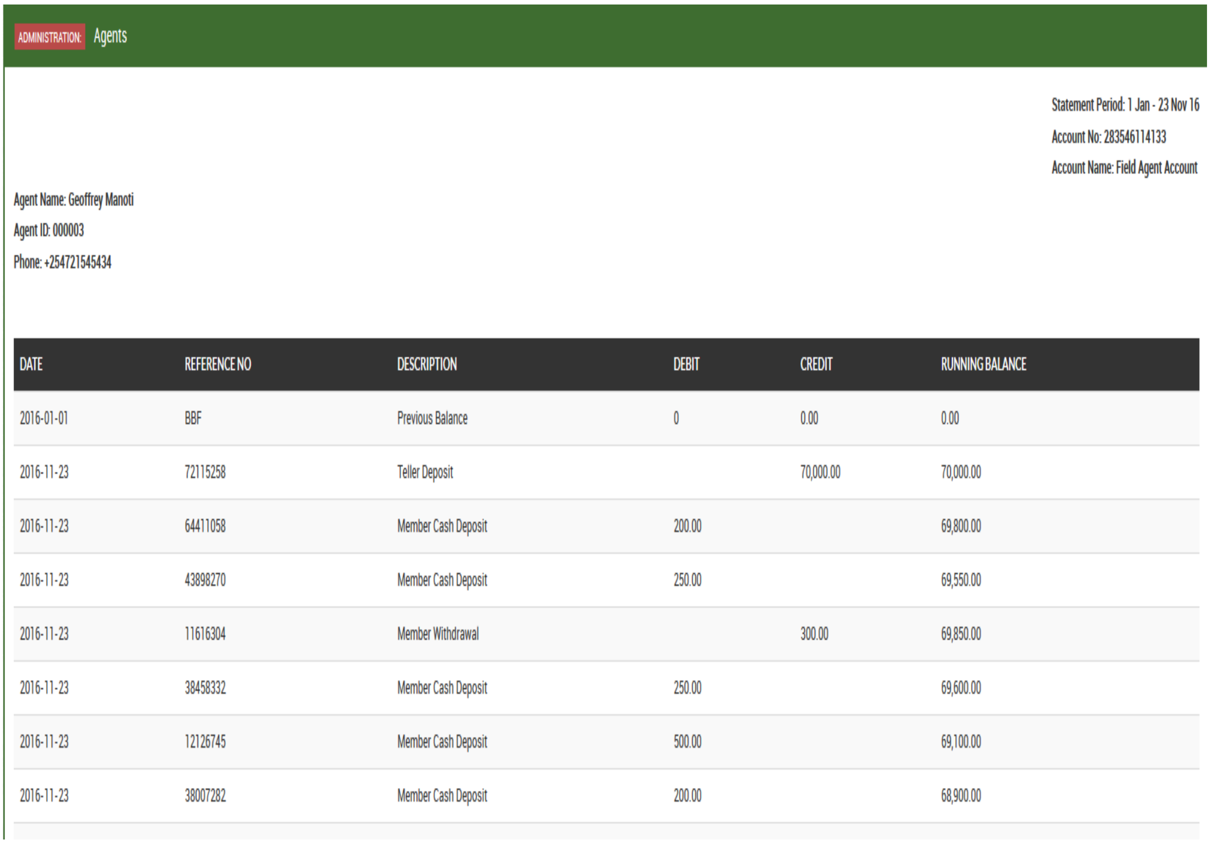
client_id	fcm01
password	56a5ec9c037f29456ae0acac1307832b0f56808d6a220d978237f721109a4111
token	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOjE2LCJpc3MiOiJodHRwczpcL1wvenVsdWJ
username	gakuya

Figure 12: Zulu Bank 1



REFERENCE #
26165495
59296493
45507995
25005160
34700486

Figure 13: Zulu Bank 2



Statement Period: 1 Jan - 23 Nov 16

Account No: 283546114133

Account Name: Field Agent Account

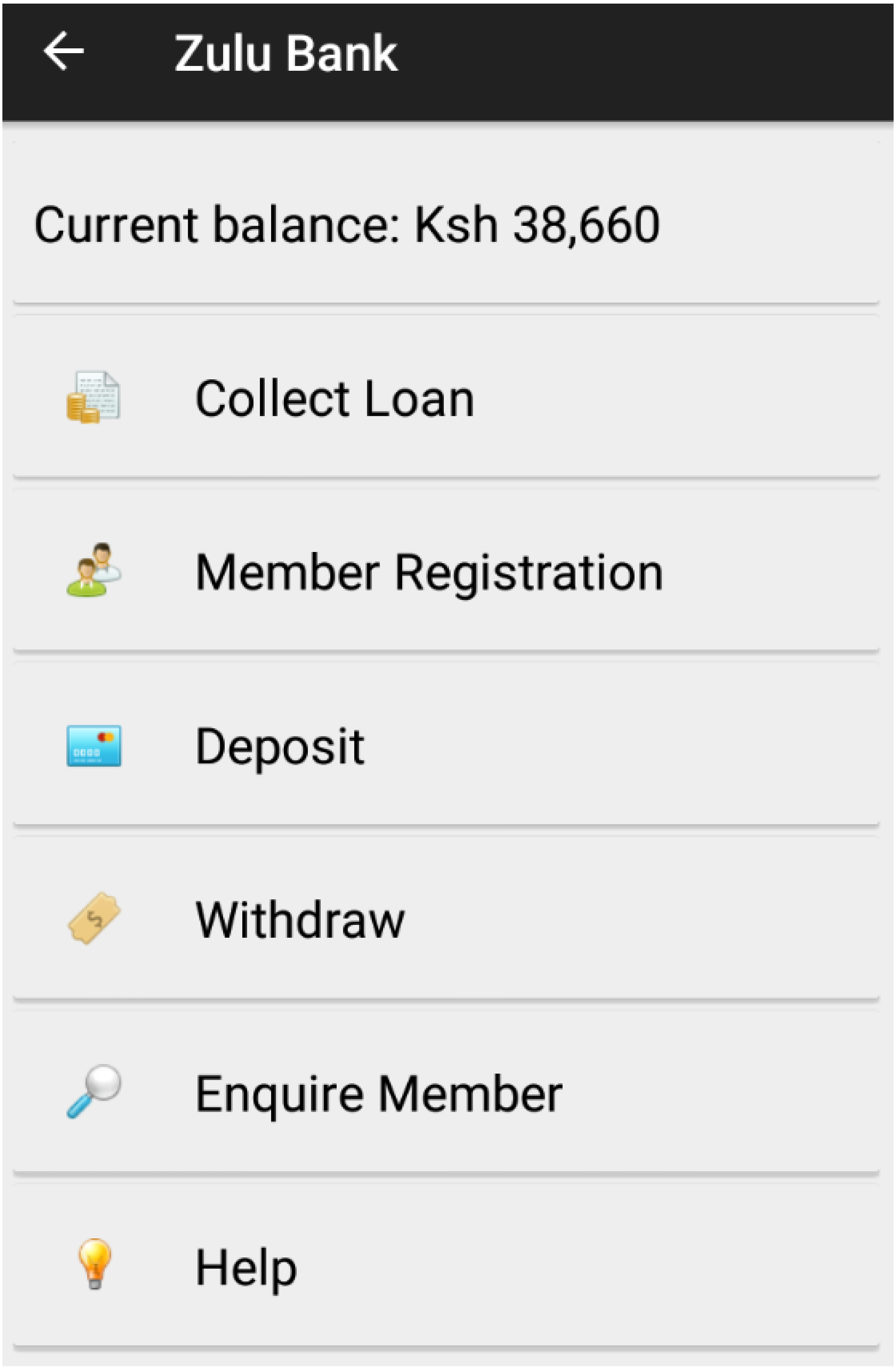
Agent Name: Geoffrey Manoti

Agent ID: 000003

Phone: +254721545434

DATE	REFERENCE NO	DESCRIPTION	DEBIT	CREDIT	RUNNING BALANCE
2016-01-01	BBF	Previous Balance	0	0.00	0.00
2016-11-23	72115258	Teller Deposit		70,000.00	70,000.00
2016-11-23	64411058	Member Cash Deposit	200.00		69,800.00
2016-11-23	43898270	Member Cash Deposit	250.00		69,550.00
2016-11-23	11616304	Member Withdrawal		300.00	69,850.00
2016-11-23	38458332	Member Cash Deposit	250.00		69,600.00
2016-11-23	12126745	Member Cash Deposit	500.00		69,100.00
2016-11-23	38007282	Member Cash Deposit	200.00		68,900.00

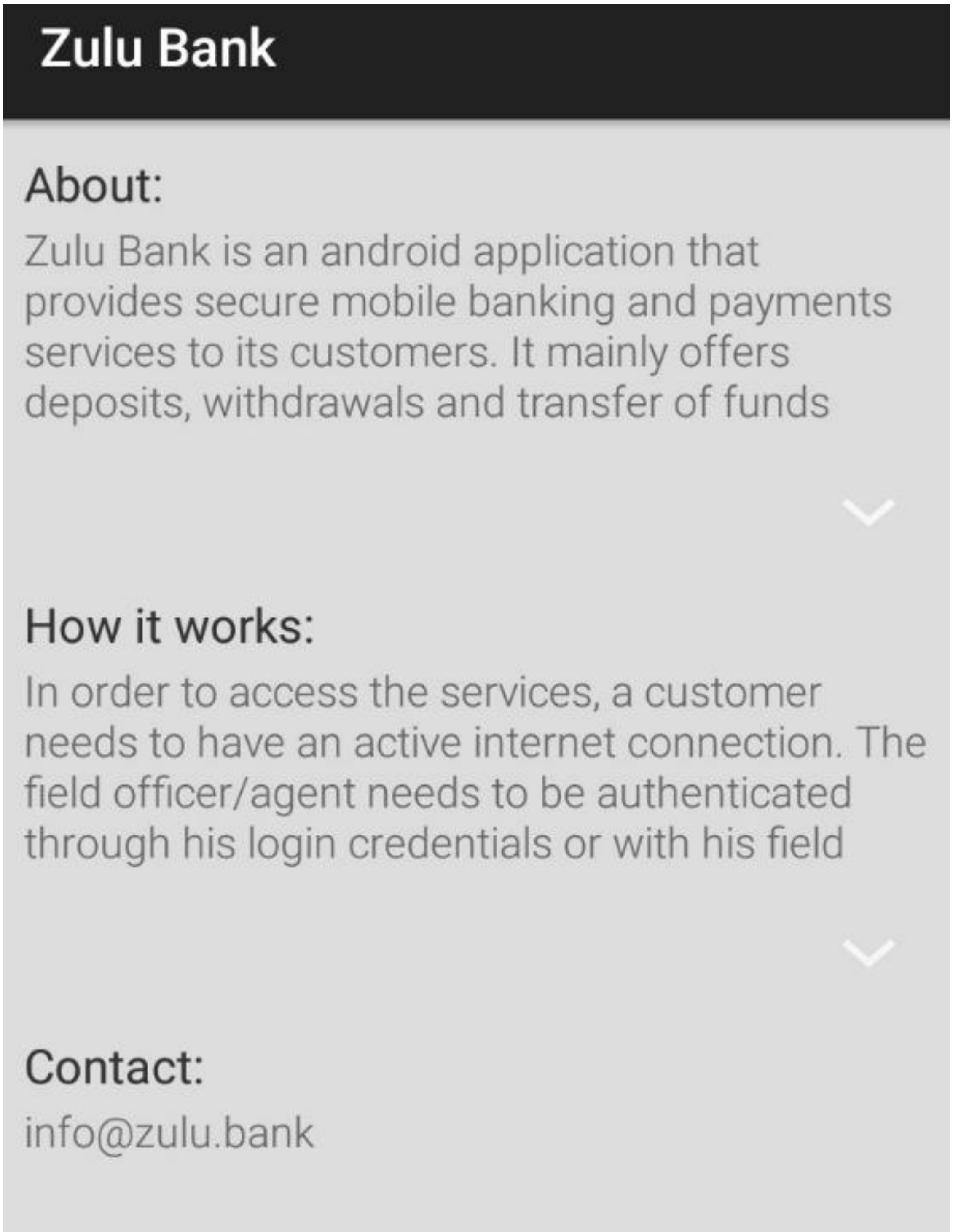
Figure 14: Zulu Bank 3



Zulu Bank

Current balance: Ksh 38,660

Figure 15: Zulu Bank Context Menu



Zulu Bank

About:

Zulu Bank is an android application that provides secure mobile banking and payments services to its customers. It mainly offers deposits, withdrawals and transfer of funds

How it works:

In order to access the services, a customer needs to have an active internet connection. The field officer/agent needs to be authenticated through his login credentials or with his field

Contact:

info@zulu.bank

Figure 16: Help

Figure 17: Kali Linux

APPENDIX 3

MOBILE BANKING & PAYMENT SECURITY SURVEY
QUESTIONNAIRE NO…………….

Introduction

My name is Geoffrey Manoti, **(Reg. No: P53/79351/2015),** a student from the University of Nairobi, School of Computing and Informatics pursuing a Master of Science degree in Distributed Computing Technology. My supervisor, Prof. W. Okelo Odongo and I, are carrying out a survey to identify and analyse the current security status for Mobile Banking and Payments in Kenya. The topic of study is **Enhancing Security of Mobile Banking and Payments in Kenya**. The data collected will be used purely for academic purposes and will be treated with the utmost confidentiality.

SECTION A: GENERAL INFORMATION

Please tick (✓) appropriately in the boxes provided below

1. Gender

A. Male [ ]

B. Female [ ]

2. Age

A. 18-25 years [ ]

B.26-35 years [ ]

C. 36-45 years [ ]

D.46-54 years [ ]

E.55 years and over [ ]

3. Highest Level of Education

A. Primary [ ]

B. Secondary [ ]

C. College [ ]

D. Undergraduate [ ]

E. Masters [ ]

F. Doctorate [ ]

4. Occupational status

A. Student [ ]

B. Employed [ ]

C. Unemployed [ ]

D. Self-employed [ ]

PART B: ACCESS TO MOBILE TECHNOLOGY

5. Do you own a mobile phone?

A. Yes [ ]

B. No [ ]

6. If yes, is it a smartphone?

A. Yes [ ]

B. No [ ]

7. Do you have a bank account?

A. Yes [ ]

B. No [ ]

8. Which service do you mostly use to carry out your bank transactions?

A. Branch Banking [ ]

B. ATM [ ]

C. Mobile Banking [ ]

D. Online Banking [ ]

9. Does your bank provide mobile banking services?

A. Yes [ ]

B. No [ ]

C. Not Sure [ ]

10. Do you use mobile banking services provided by your bank?

A. Yes [ ]

B. No [ ]

11. If yes, what is the name of the bank? ………………………..

12. If your response in Q10 was yes, for how long have you used Mobile Banking services?

A. Less than 1-year [ ]

B. 1 – 5 years [ ]

C. 6 – 10 years [ ]

SECTION C: SECURITY OF MOBILE BANKING

No.	Please indicate the extent to which you agree with the following statements on Mobile Banking and Payments, on a five point Likert scale (1-5). **Please tick (✓) appropriately in the boxes provided.**	Strongly Agree	Agree	Don't know	Disagree	Strongly Disagree
1.	**MOBILE BANKING SECURITY/PRIVACY**	**5**	**4**	**3**	**2**	**1**
a.	Using mobile banking is not secure.					
b.	Security of mobile banking is wanting.					
c.	Do you consider security as a concern when using mobile banking?					
d.	Banks do not have the ability in mobile banking to protect my privacy.					
e.	Matters of security have an influence on my using mobile banking.					
No.	Please indicate the extent to which you agree with the following statements on Mobile Banking and Payments, on a five point Likert scale (1-5). **Please tick (✓) appropriately in the boxes provided.**	Strongly Agree	Agree	Don't know	Disagree	Strongly Disagree
2.	**MOBILE BANKING AVAILABILITY/RELIABILITY**	**5**	**4**	**3**	**2**	**1**
a.	Poor network connectivity affects mobile banking transactions.					
b.	Reliability is a factor influencing my use of mobile banking.					
c.	Mobile banking enables me to access banking services more quickly and enhances effectiveness.					
d.	Mobile banking service is useful to me.					
e.	Mobile banking is easy to use.					
f.	Instructions for using mobile banking are easy to follow.					

3. The below factors affect the use/adoption of Mobile Banking. Weigh them on a scale of 1-5 according to their importance. **Please tick (✓) appropriately in the boxes provided below.**	Extremely important	Very important	Quite important	Somewhat important	Unimportant
	5	**4**	**3**	**2**	**1**
Security.					
Privacy.					
Reliability.					
Cost.					
Accessibility and ease of use.					

4.	Below are issues that can be experienced when accessing Mobile Banking and Payments services. Please indicate the extent to which you agree with the issues below, on a five point Likert scale (1-5). **Please tick (✓) appropriately in the boxes provided.**	Strongly Agree	Agree	Don’t Know	Disagree	Strongly Disagree
		5	**4**	**3**	**2**	**1**
a.	Problems while connecting to the Mobile Banking service via USSD (Unstructured Supplementary Service Data). e.g. *144#					
b.	Problems while trying to connect to the Mobile Banking service via a Mobile Banking application.					
c.	Lack of enough airtime to carry out transactions through the Mobile Service Provider, when accessing Mobile Banking services.					
d.	Network problem while trying to connect to the Mobile Banking service.					
e.	Mobile Banking application errors.					
f.	Delay in performing bank transactions. .e.g. Funds transfer, deposits and withdrawals.					