Journal of Education and Science
Vol. 27, No.3, 2018, pp. 120-135
ISSN 1812-125X
http://edusj.mosuljournals.com

SMS-Phishing on Android Smart Phone

MAHMOD S. MAHMOD

College of Science, University of Mosul, Iraq
Corresponding Author: mahmoodsubhy1981@gmail.com

تاريخ الاستلام 2014/01/28 تاريخ القبول 2014/06/04

Abstract

Recently, mobile become the most significant device that used almost daily, and one of the most important service that offered by mobile operators is the SMS that used between mobiles, none of the mobile OS is contain a service that encrypt the send SMS where some needs to send an important, personal or secure message to the other mobile, and in this case mobiles OS needs an application to encrypt the SMS and then decrypted it on the other side. From this conception the idea of this research become to develop an application working on the devices that supports Android environment, this application encrypt the SMS before it sends from sender and then decrypt it by using the same application when it arrives to the recipient.

The goal from Using the encryption Operation of the SMS is to convince that the user is using a secure SMS application that it sends an encrypted SMS where in fact that is not the truth, but the application find the loophole to take the encrypted SMS and sends the encrypted SMS to the third party (hacker) where the user will not noticed that the SMS went to the third party. Java language and an android developer tools like (ECLIPS) and other tools where used to build this application. This application is working on android environment and also needs a little space of memory and it works without leaves anything that refers to the third-party.

Keywords: *Android, Operating System, Mobile device, text message, encryption process, Eclipse .*

Introduction

From large enterprises and government agencies to small businesses and consumers, the use of smart phones and other mobile devices to manage professional and personal interactions is now ubiquitous. Mobile devices have become the new personal computer, storing as much data as a PC but providing greater flexibility and portability. Online banking, commerce, and other business applications put daily business and financial transactions at users' fingertips. And, at every turn, users are implored to download productivity and entertainment applications to further increase the value of their mobile devices[1] .

Phishing and spam attacks are two of the most successful and profitable attacks [2]. Both attacks have achieved great success through traditional medias, such as email, web, and instant messaging.

With the fast growth of smart phone markets, we have seen more and more attempts to launch phishing and spam attacks on popular smart phone platforms[3] .

While smart phones and tablet devices now perform the same functions as a PC, one critical feature is missing— security. Whereas most PCs come equipped with antivirus and other endpoint security software, the vast majority of mobile devices are devoid of any security protection, leaving both the data and applications on these mobile devices at risk of exploitation or misuse[1] .

The increasing number of mobile-related exploits, and the growing impact of these security breaches combined with the continuing exponential growth in mobile devices sold and in use, have put the mobile industry at a critical juncture: mobile security *must* be addressed in 2011 in order to ensure the privacy and safety of users' critical personal and business information and data[1].

1- Related Works

We present some of the relevant literature for Phishing attacks on Android and iOS platforms, Zhi Xu, Sencun Zhu in (2012) studied that notification customization may allow an installed Trojan application to launch phishing attacks or anonymously post spam notifications. Through there studies on four major Smartphone platforms, they show that both Android and BlackBerry OS are vulnerable under the phishing and spam notification attacks .

iOS and Windows Phone allow little notification customization, thus launching the phishing and spam attacks will expose the identity of the Trojan application . Attack demonstrations on all platforms are presented [3].

Adrienne Porter Felt, David Wagner in 2012 also implementation of sample phishing attacks on the Android and iOS platforms demonstrates

that attackers can spoof legitimate applications with high accuracy, suggesting that the risk of phishing attacks on mobile platforms is greater than has previously been appreciated [4]. *Min Wu* in 2005 provide Thesis Proposal: Fighting Phishing at the User Interface [5] . In 2012 SCB provide Example of a phishing SMS sent to Android smart phones with an embedded link for Trojan program installation in an attempt to steal SCB Easy Net usernames and passwords [6] .

2- Phishing and Phishing Techniques

3-1-What is phishing?

Phishing is a fraudulent attempt, usually made through email, to steal your personal information. The best way to protect yourself from phishing is to learn how to recognize a phish[7].

Phishing emails usually appear to come from a well-known organization and ask for your personal information — such as credit card number, social security number, account number or password. Often times phishing attempts appear to come from sites, services and companies with which you do not even have an account.

In order for Internet criminals to successfully "phish" your personal information, they must get you to go from an email to a website. Phishing emails will almost always tell you to click a link that takes you to a site where your personal information is requested. Legitimate organizations would never request this information of you via email.

3-2-Phishing Techniques

There are a number of different phishing techniques used to obtain personal information from users. As technology becomes more advanced, the phishing techniques being used are also more advanced. To prevent Internet phishing, users should have knowledge of various types of phishing techniques and they should also be aware of anti-phishing techniques to protect themselves from getting phished. Let's look at some of these phishing techniques[8].

1- Email / Spam .
2-Web Based Delivery .
3-Instant Messaging .
4- Trojan Hosts .
5- Link Manipulation .
6- Key Loggers .
7- Session Hacking .
8- System Reconfiguration .
9- Content Injection .
10- Phishing through Search Engines

11- Phone Phishing
12- Malware Phishing

In recent years, a new type of phishing called Smishing is a combination of the words SMS and phishing. The former stands for "Short Message Service" and is the official term for text messages on your cellular phone[9]. From the above, it follows that smishing is nothing but a phishing attempt using the phone as a medium as opposed to a computer. Social engineering is used to get a phone user to do the scammers' bidding. Some examples from Wikipedia:

- Notice - this is an automated message from (a local credit union), your ATM card has been suspended. To reactivate call urgent at 866-###-####
- We're confirming you've signed up for our dating service. You will be charged $2/day unless you cancel your order on this URL: www.?????.com.
- (Name of popular online bank) is confirming that you have purchase a $1500 computer from (name of popular computer company). Visit www.?????.com if you did not make this online purchase
- (Name of a financial institution): Your account has been suspended. Call 702.354.0818 immediately to reactivate

3- Android And Development Android Application

4-1 What Is Android?

Android is a mobile operating system that is based on a modified version of Linux. It was originally developed by a startup of the same name, Android, Inc. In 2005, as part of its strategy to enter the mobile space, Google purchased Android and took over its development work (as well as its development team) .

Google wanted Android to be open and free; hence, most of the Android code was released under the open-source Apache License, which means that anyone who wants to use Android can do so by downloading the full Android source code [9].

4-2 Tools for Development Android Application

4-2-1 Eclipse

The first step towards developing any applications is obtaining the integrated development environment (IDE). In the case of Android, the recommended IDE is Eclipse, a multi-language software development environment featuring an extensible plug-in system. It can be used to develop various types of applications, using languages such as Java, Ada, C, C++, COBOL, Python, etc.

For Android development, you should download the Eclipse IDE for Java EE Developers(www.eclipse.org/downloads/packages/eclipse-ide-java-eedevelopers/ heliossr1). Six editions are available: Windows (32 and 64-bit), Mac OS X (Cocoa 32 and 64), and Linux (32 and 64-bit). Simply select the relevant one for your operating system. All the examples in this book were tested using the 32-bit version of Eclipse for Windows.

4-2-2 Android SDK

The next important piece of software you need to download is, of course, the Android SDK. The Android SDK contains a debugger, libraries, an emulator, documentation, sample code, and tutorials. You can download the Android SDK from http://developer.android.com/sdk/index.html.

4-2-3 Android Development Tools (ADT)

The Android Development Tools (ADT) plug-in for Eclipse is an extension to the Eclipse IDE that supports the creation and debugging of Android applications. Using the ADT, you will be able to do the following in Eclipse:

- ➤➤ Create new Android application projects.
- ➤➤ Access the tools for accessing your Android emulators and devices.
- ➤➤ Compile and debug Android applications.
- ➤➤ Export Android applications into Android Packages (APK).
- ➤➤ Create digital certificates for code-signing your APK.

4-3 Android Permissions

In order to protect Android users, applications' access to phone resources is restricted with permissions. An application must obtain permissions in order to use sensitive resources like the camera, microphone, or call log. For example, an application must have the READ_CONTACTS permission in order to read entries in a user's phonebook. Android 2.2 defines 134 permissions.

Obtaining permissions is a two-step process. First, an application developer declares that his or her application requires certain permissions in a file that is packaged with the application. Second, the user must approve the permissions requested before installation.

Each application has its own set of permissions that reflects its Figure 1: On the left, a screenshot of the Android Market's final installation page, displaying the application's permission requests. On the right, the permission dialog that appears if a user clicks on a permission warning.

The official Android Market provides every application with two installation pages. The first installation page includes a description, user

reviews, screenshots, and a "Download" button. After pressing "Download," the user arrives at a final installation page that includes the application's requested permissions (Figure 1) [11] .

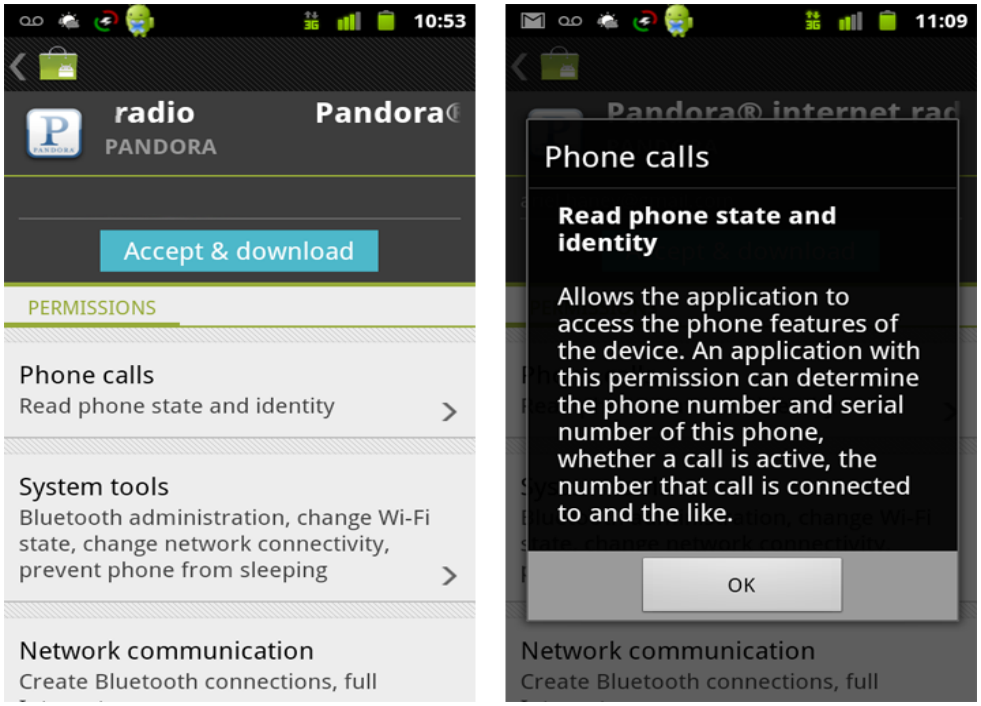

Figure 1: On the left, a screenshot of the Android Market's final installation page, displaying the application's permission requests. On the right, the permission dialog that appears if a user clicks on a permission warning.

4-3-1 Introducing Permissions

Permissions are an application-level security mechanism that lets you restrict access to application components. Permissions are used to prevent malicious applications from corrupting data, gaining access to sensitive information, or making excessive (or unauthorized) use of hardware resources or external communication channels.

many of Android's native components have permission requirements. The native permission strings used by native Android Activities and Services can be found as static constants in the android.Manifest.permission class.

To use permission-protected components, you need to add <uses-permission> tags to application manifests, specifying the permission string that each application requires, When an application package is installed, the permissions requested in its manifest are analyzed and granted (or denied) by checks with trusted authorities and user feedback.

Unlike many existing mobile platforms, all Android permission checks are done at installation. Once an application is installed, the user will not be prompted to reevaluate those permissions[12].

4-3-2 Declaring and Enforcing Permissions

Before you can assign a permission to an application component, you need to define it within your manifest using the <permission> tag as shown in the Listing 1 [12] .

LISTING 1 : Declaring a new permission

```
<permission
android:name="com.paad.DETONATE_DEVICE"
android:protectionLevel="dangerous"
android:label="Self Destruct"
android:description="@string/detonate_description">
</permission>
```

Within the permission tag, you can specify the level of access that the permission will permit (normal , dangerous, signature, signature Or System), a label, and an external resource containing the description that explains the risks of granting this permission.
In each case, you can add a permission attribute to the application component in the manifest, specifying a required permission string to access each component.
Listing 2 shows a manifest excerpt that requires the permission defined in Listing 1 to start an Activity.

LISTING 2 : Enforcing a permission requirement for an Activity

```
<activity
android:name=".MyActivity"
android:label="@string/app_name"
android:permission="com.paad.DETONATE_DEVICE">
</activity>
```

Content Providers let you set readPermission and writePermission attributes to offer a more granular
control over read/write access.

4- SmS messaging

SMS messaging is one of the main *killer applications* on a mobile phone today — for some users as necessary as the phone itself. Any mobile phone you buy today should have at least SMS messaging capabilities, and nearly all users of any age know how to send and receive such messages. Android comes with a built-in SMS application that enables you to send and receive SMS messages. However, in some cases you might want to integrate SMS

capabilities into your own Android application. For example, you might want to write an application that automatically sends a SMS message at regular time intervals. For example, this would be useful if you wanted to track the location of your kids — simply give them an Android device that sends out an SMS message containing its geographical location every 30 minutes[10].

5-1 Sending SMS Messages Programmatically

Using this approach, your application can automatically send an SMS message to a recipient without user intervention.

Android uses a permissions-based policy whereby all the permissions needed by an application must be specified in the AndroidManifest.xml file. This ensures that when the application is installed, the user knows exactly which access permissions it requires.
Because sending SMS messages incurs additional costs on the user's end, indicating the SMS permissions in the AndroidManifest.xml file enables users to decide whether to allow the application to install or not.
To send an SMS message programmatically, you use the SmsManager class. Unlike other classes, you do not directly instantiate this class; instead, you call the getDefault() static method to obtain a SmsManager object. You then send the SMS message using the sendTextMessage() method:

```
private void sendSMS(String phoneNumber, String message)
{
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(phoneNumber, null, message, null, null);
}
```

Following are the five arguments to the sendTextMessage() method:

➤➤ destinationAddress — Phone number of the recipient
➤➤ scAddress — Service center address; use null for default SMSC
➤➤ text — Content of the SMS message
➤➤ sentIntent — Pending intent to invoke when the message is sent
➤➤ deliveryIntent — Pending intent to invoke when the message has been delivered.

5-2 Sending SMS Messages Using Intent

Using the SmsManager class, you can send SMS messages from within your application without the need to involve the built-in Messaging application. However, sometimes it would be easier if you could simply

invoke the built-in Messaging application and let it do all the work of sending the message.
To activate the built-in Messaging application from within your application, you can use an Intent object together with the MIME type "vnd.android-dir/mms-sms" as shown by the following code snippet:

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
btnSendSMS = (Button) findViewById(R.id.btnSendSMS);
btnSendSMS.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
//sendSMS("5556", "Hello my friends!");
Intent i = new
Intent(android.content.Intent.ACTION_VIEW);}
i.putExtra("address", "5556; 5558; 5560");}
i.putExtra("sms_body", "Hello my friends!");}
i.setType("vnd.android-dir/mms-sms");}
startActivity(i);}
```

5-3 receiving SMS messages

Besides sending SMS messages from your Android applications, you can also receive incoming SMS messages from within your application by using a BroadcastReceiver object. This is useful when you want your application to perform an action when a certain SMS message is received. For example, you might want to track the location of your phone in case it is lost or stolen. In this case, you can write an application that automatically listens for SMS messages containing some secret code. Once that message is received, you can then send an SMS message containing the location's coordinates back to the sender.

To listen for incoming SMS messages, you create a BroadcastReceiver class. The BroadcastReceiver class enables your application to receive intents sent by other applications using the sendBroadcast() method. Essentially, it enables your application to handle events raised by other applications. When an intent is received, the onReceive() method is called; hence, you need to override this.
When an incoming SMS message is received, the onReceive() method is fired. The SMS message is contained in the Intent object (intent; the second parameter in the onReceive() method) via a Bundle object. The messages

are stored in an Object array in the PDU format. To extract each message, you use the static createFromPdu() method from the SmsMessage class.

One interesting characteristic of the BroadcastReceiver is that you can continue to listen for incoming SMS messages even if the application is not running; as long as the application is installed on the device, any incoming SMS messages will be received by the application.

5-4 Invoking an Activity from a BroadcastReceiver

The previous Sections shows how you can pass the SMS message received to be displayed in the activity.
However, in many situations your activity may be in the background when the SMS message is received. In this case, it would be useful to be able to bring the activity to the foreground when a message is received. ” as shown by the following code snippet

```
@Override
public void onReceive(Context context, Intent intent)
{
//---get the SMS message passed in---
Bundle bundle = intent.getExtras();
SmsMessage[] msgs = null;
String str = “”;
if (bundle != null)
{
//---retrieve the SMS message received---
Object[] pdus = (Object[]) bundle.get(“pdus”);
msgs = new SmsMessage[pdus.length];
for (int i=0; i<msgs.length; i++){
msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
str += “SMS from “ + msgs[i].getOriginatingAddress();
str += “ :”;
str += msgs[i].getMessageBody().toString();
str += “\n”;
}
//---display the new SMS message---
Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
//---launch the MainActivity---
Intent mainActivityIntent = new Intent(context, MainActivity.class);
mainActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
context.startActivity(mainActivityIntent);
//---send a broadcast to update the SMS received in the activity---
Intent broadcastIntent = new Intent();
broadcastIntent.setAction(“SMS_RECEIVED_ACTION”);
broadcastIntent.putExtra(“sms”, str);
context.sendBroadcast(broadcastIntent);
```

5- Implementation of Application

The Application is designed using Android development Tools (Eclipse) with 32-bit and Java v7.5 language. the Application consists of a single Activity (Interface) which performs three operations sending,

receiving messages and Listening to incoming messages, as shown in Figure (2) .

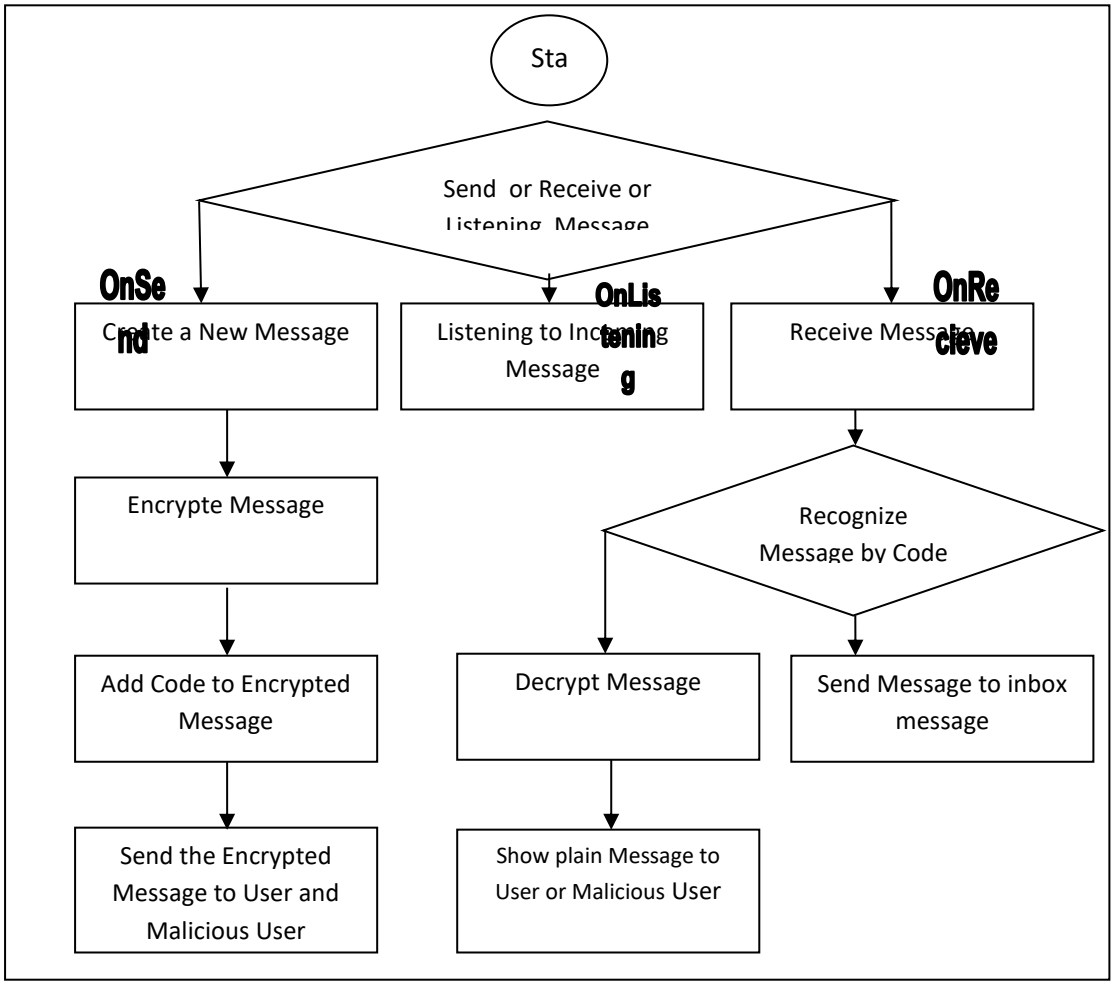

Figure 2: design of Application

Sending operation starts by placing a new message in a buffer and then passing the message to encrypt process for encrypting the message using simple encryption method (by replacing the characters in the odd position with the characters in the even position) because it need a few space in memory as well as reduce the time of encryption process. then add a special code to the beginning of the encrypted message, which used to distinguish between the messages belong to Application and the normal messages in the recipient side after that send the encrypted message to the two sides at the same time, the first to the recipient and the second to malicious user.

After installing the application on Mobile Device, the application works to listening on received SMS messages by the device where this operation running in the background.

If the device receive a SMS message the application recognize the message belong to it or not by checking the code where as the message not belong to it then forward message to inbox (the message belong to bulit_in

Application on Android) otherwise the Application Decoded and display the message to the recipient side and spam side at the same time .

6- How Application Work

In order to be a phishing successfully should installation the Application in the devices sent (the victim) Device, received Device and spam Device, where in the installation process will ask the user to see the permissions that the Application uses it one of them send messages that appear to the user Normally because the Application works to send encrypted messages and needs to those permissions, as shown in Figure (3), then choose install to install the Application on the device.

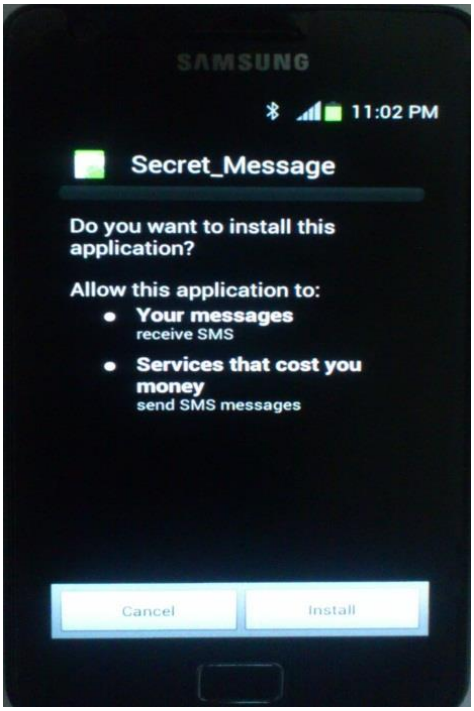

Figure (3) shows permissions used by the Application

Figure (4) shows a recipient User (phone no. 5554) and a spam User (phone no. 5556) at the beginning of Work .

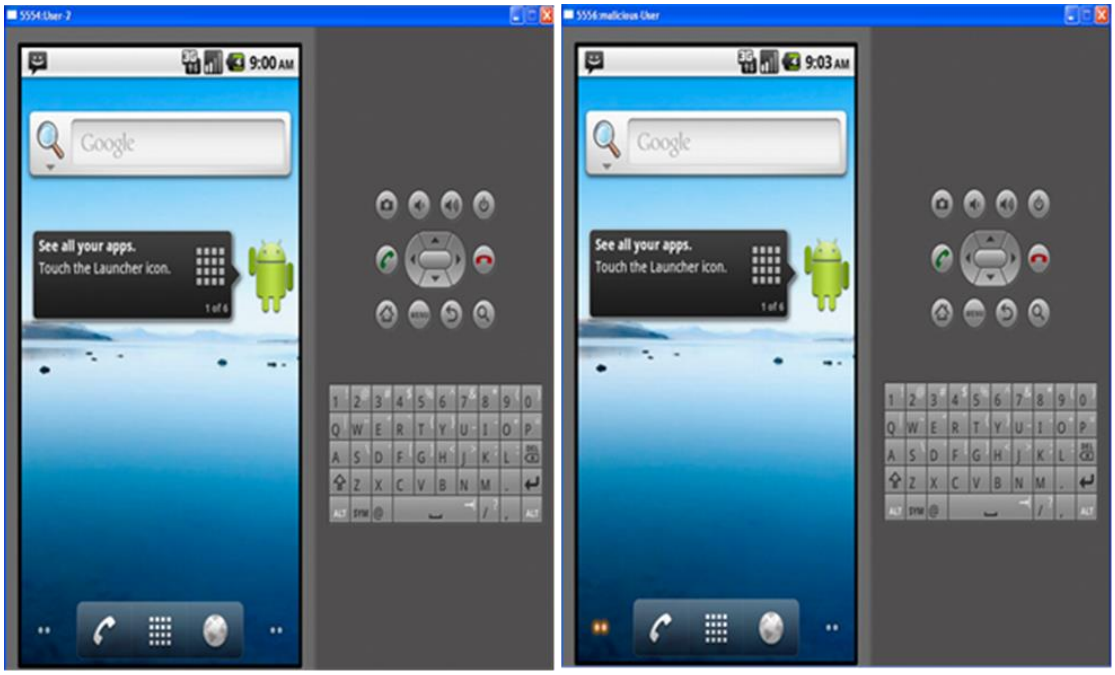

Figure (4) shows a recipient User and a spam User at the beginning of Work .

After Install Application in Sender User Device and Play the Application will show the Interface as in the figure (5) .

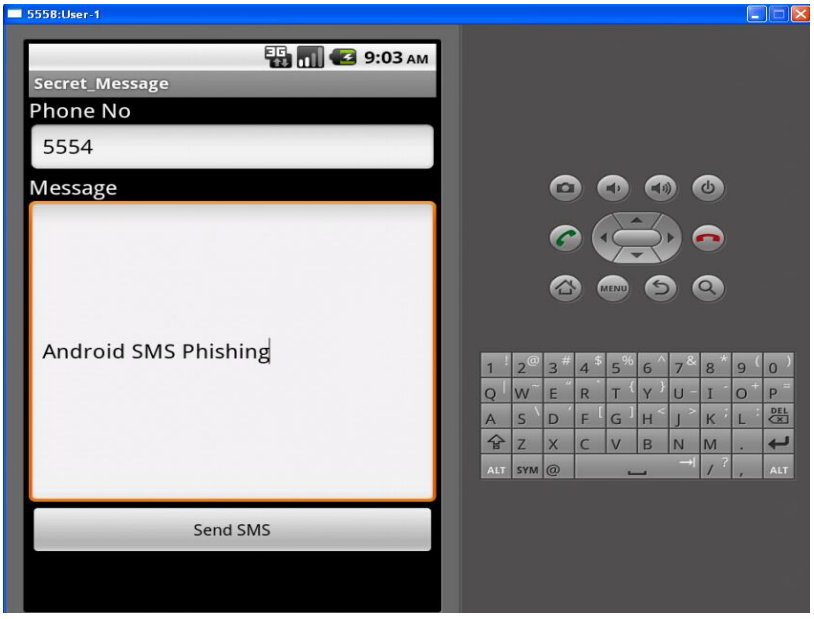

Figure (5) show the Interface of Application

put phone no for recipient in textbox labeled as (Phone No) and put the plain message in textbox labeled as (Message) then press the button (Send SMS), the Application work to encrypt plain message and add the Special code for it and send the encrypted message to recipient User and Malicious User at the same time as show in the figure (6) .

Note : when then Application send encrypted message, the encrypted message not appear in the box for sending message hence the sender User not known that a copy form encrypted message go to Malicious User.

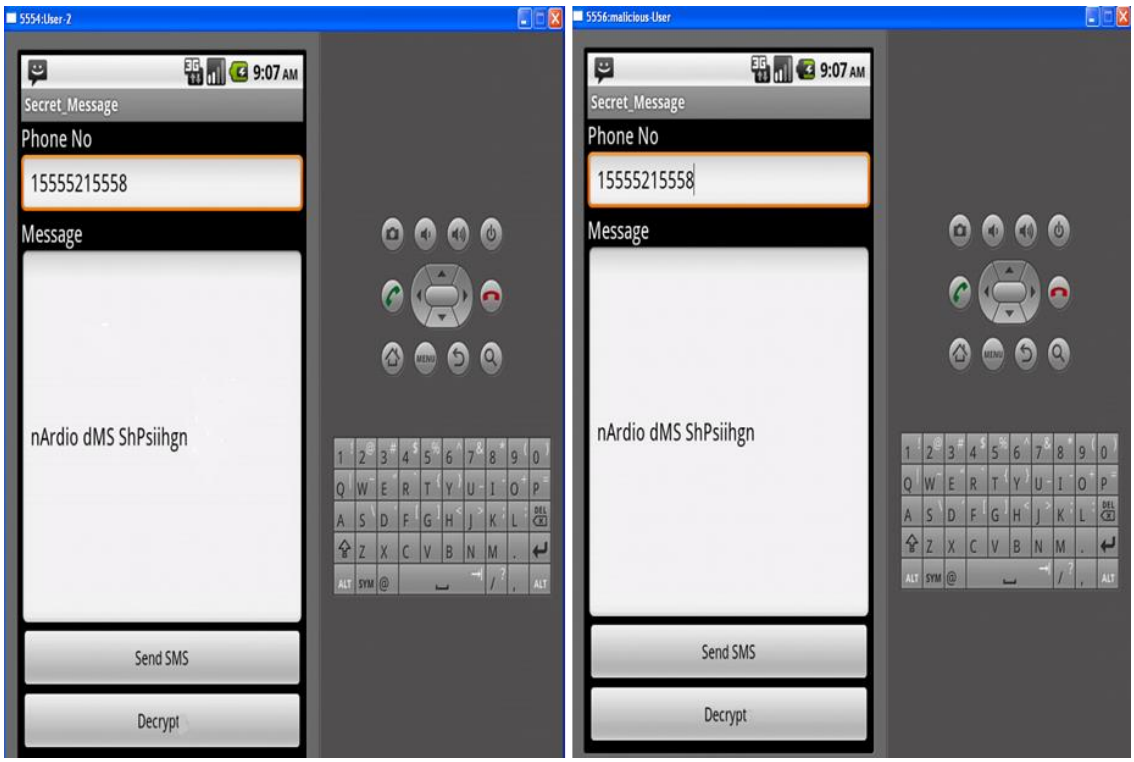

Figure (6) show the interface of Application when received message

When the encrypted message received by device and the Application recognize it by the special code, it open the Interface for Application directly and put the Sender phone No. in textbox Labeled (phone No) and put the Encrypted message in textbox Labeled (Message) and visible the button labeled (Decrypt) as show in the figure (6), when (receiver User or Malicious User) press the button labeled (Decrypt), the Application make to decrypt the encrypted message and show the result in textbox labeled (Message) as show in the figure (7) .

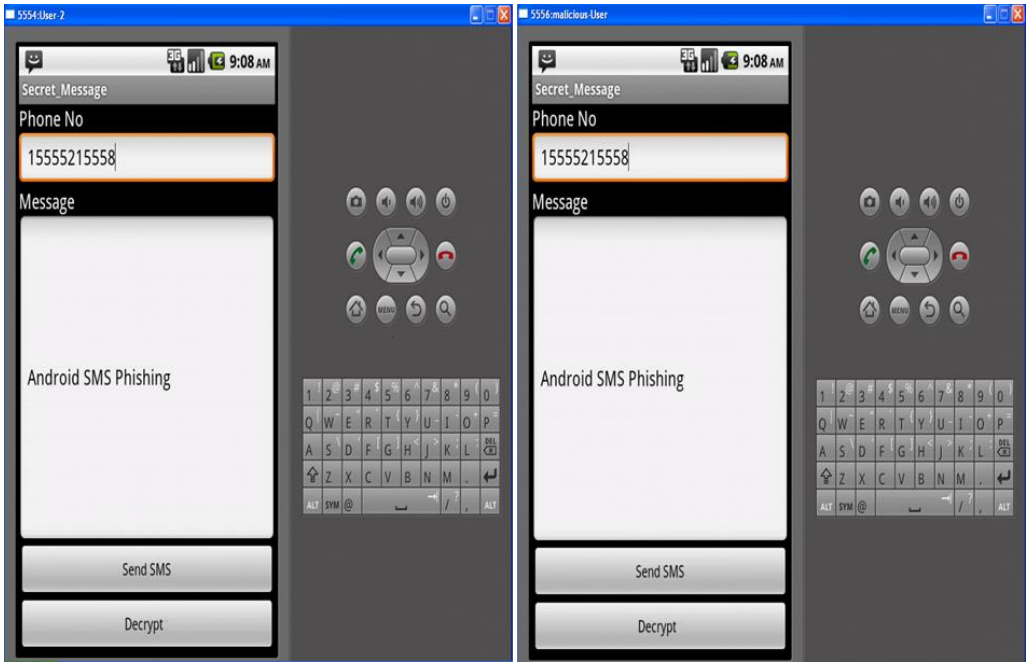

Figure (7) show the result of decrypt encrypted message

7- Conclusion and future Work

Phishing information Began in the Internet where they are getting mail messages as well as passwords for user accounts and special information stored in computers and servers and network devices using many of Phishing Techniques at the same time several software and physical hardware was developed to detect and block attempts to computer via local networks and Internet. In recent years, attempts moved into the mobile phone networks where software has been developed on the potential phishing information contained within Mobile phone such as a list of names and passwords and other Special Information whether mobile connected to the internet or offline , and this software a Published in several locations on the Internet**,** This shows that the Android operating system contains many loopholes that can be exploited by a malicious user to work spambots, so the android operating system needs to intrusion detection systems as well as programs detect viruses .

In this research New Application has been designed based on Phishing text messaging (SMSPhishing) works under the Android operating system environment, where the Application send a copy of the text messages (sent from the first party to the second party) to a third party without feeling the first party that copy of text message send to the malicious User, using the encryption process to trick the user to downloading the Application on Mobile device and Install it because when install the Application on Mobile will prompt the user for approval of the

permittivity send text messages as the Application to send text messages and encrypted manner . Thus, start from this idea in future an researcher can design a Application to send text messages from a third party (malicious User) as to come from the first party to the second party and this idea need to know the basics of business and control units in mobile networks in addition to the analysis of text messages fields.

References

[1] Juniper Networks Global Threat Center Research (2011). "Malici ous Mobile Threats Report 2010/2011".

[2] R. Dhamija; J. D. Tygar; M. Hearst (2006). "Why phishing works" in Proc. of the SIGCHI conference on Human Factors in computing systems .

[3] Zhi Xu; Sencun Zhu, (2012). "Abusing Notification Services on Smartphones for Phishing and Spamming", Department of Computer Science & Engineering, Pennsylvania State University .

[4] Adrienne Porter Felt; David Wagner, (2012). " Phishing on Mobile Devices " University of California, Berkeley .

[5] Min Wu, (2005). "Thesis Proposal: Fighting Phishing at the User Interface" .

[6] SCB,(2012). " Example of a phishing SMS sent to Android smart phones with anembedded link for Trojan program installation in an attempt to steal SCB Easy Net usernames and passwords ", http://scb.kcyberbank info/scbeasy.apk .

[7] http://www.phishtank.com/what_is_phishing.php .

[8] http://www.phishing.org/phishing-techniques/ .

[9]http://www.alertboot.com/blog/blogs/endpoint_security/archive/2012/11/14/ smishing-sms-phishing-present-and-on-the-rise-on-android.aspx.

[10] Wei-Meng Lee, (2011). "Beginning Android Application Development", ISBN: 978-1-118-01711-1.

[11] Adrienne Porter Felt,eta, (2012)." Android Permissions: User Attention, Comprehension, and Behavior", Computer Science Department, University of California, Berkeley .

[12] Reto Meier, (2010)." Professional Android™ 2 Application Development", ISBN: 978-0-470-56552-0 .