CS 161
Computer Security

Dawn Song
Fall 2012

Lab 6
Android Malware

Due Date: Dec 5, 2012 by 11:59pm

1 Background

In this lab you will be tasked with analyzing an infected Android system. The purpose of the lab will be to familiarize with you with the inner workings of Android applications and the way in which applications interact with one another through Intents.

2 Overview

In this lab, you will be given a VirtualBox image of an infected phone. Starting from this image, you need to identify the malicious application, analyze its behaviors and exploit a vulnerability in the malicious application. To get started, do the Notepad tutorial to make sure you have a basic understanding of:

1. How an Android application is developed
2. How different components communicate with each other
3. How the Android permission system works.

3 Setting up the Development Environment

For this lab you will be provided a Virtual Machine image of an Android device and Ubuntu Linux system (`cs161:cs161-lab6`). The Ubuntu system will have most of the required tools to complete the lab (you still need to install the decompilation tools yourself though). Alternatively you can install the Android SDK and the required Apk Decompilation tools on your own machine. The VM of the Android Device is necessary to complete this lab. Both VMs require Virtual Box or VMware (whichever one you prefer). All the necessary files for this lab can be downloaded from `http://www.eecs.berkeley.edu/~mor/cs161-lab6`.

3.1 Running Android in VirtualBox

1. Download the image and run it inside VirtualBox.
2. Start up the VM, and wait for the boot process to complete
3. Enter the Android Console with `Alt+F1` (those of you on OS X may need to do `Fn+Option+F1`)
4. Obtain the IP address of the VM by typing `netcfg` and noting the address for eth0. You will use this IP to connect to the VM.
5. Return to the normal interface with `Alt+F7`. If you encounter problems with the virtual machine see section 5 below.

3.2 ADB

Connect to the Android virtual machine by running `adb connect [IP]` from the provided Ubuntu Linux image or your host machine. [IP] is the IP address you obtained above. Before running any ant commands, ensure you are connected to the VM with `adb devices`.

3.3 Decompilation Tools on Host Machine or VM

1. Download and extract **dex2jar**
2. Download and extract **jd-gui** for your platform
3. Download and install **android-apktool** using the instructions provided on their main page.

3.4 Installing Android SDK on Host Machine (Already installed on VM)

1. **Download and Install the Android SDK**
2. **Install Android 2.3.3 SDK**

4 Problems

4.1 Find the malicious app (80 points)

For the first step, you need to find out which installed application is the malicious one. Android keeps all the installed applications as *.apk files in the /data/app folder. To find the malicious application, you will need to use adb to enter VM and extract the necessary apk files. Follow the instructions above to connect adb to the VM. Use the ADB Reference to extract the apk files.

Once you have extracted the applications, find the malicious apk file by examining the AndroidManifest.xml files for the permissions they request and the intent-filters they register. To access the inside of the APK, a short little guide is available at DroidDudes. Once you locate a few suspicious candidates, you need to take a brief look at the code to confirm the actual malicious one.

Question 1

Triggers: A common behavior of traditional PC malware is to make themselves "autostartable". Similarly, this mobile malware does this as well. It's very important to identify these entry points so we know how malicious behaviors are triggered. Identify the configuration in the manifest file that makes the malware autostartable.

Question 2

System APIs As a first step towards malicious code analysis, we can get a rough idea of what this malicious code can do by dumping system API calls. Here system APIs refer to any methods that is called but not implemented in the application (namely the apk file) itself:

```
invoke-virtual {v1, v2} \
Lorg/apache/http/impl/client/DefaultHttpClient;\
->execute(...
```

List all the system APIs you can find in the malicious code that retrieves user's private data or the phone's information.

Question 3

List all the system API names you can find in the malicious code that can send data to remote servers.

Question 4

Locate Remote Server: Having a rough idea of what this malware can do, now we need to carefully read its source code and figure out more precisely what it does. By looking at the code, find the URL of the server that the malicious code is talking to over Internet.

Question 5

Command format: The botnet client keeps retrieving new commands from a remote server. By analyzing the code, figure out in what format these commands are passed.

Question 6

Commands: What commands are supported in this malicious client? Write down all the commands with a short description for each of them.

4.2 Secure Messaging (30 points)

For this section, you will complete an application that communicates with a background network service through intents. The primary objective of this section is to familiarize yourself with the Android Intent message passing system. Examine the provided Android Manifest and `StudentMessageService.java` to determine the correct means by which to send intents to this service. The following resources will be useful to you.

Android Manifest
Intents and Intent Filters

The skeleton of the application is in directory `studentMessage`. **Note:** You are only allowed to modify the appropriate sections `StudentMessageActivity.java` to send intents.

To prepare the build environment, make sure both [SDK]/platform-tools and [SDK]/tools are in your PATH environment variable, and then type `android update project -p . --target android-10` in the `studentMessage` directory. To compile and install `studentMessage`, type `ant debug install` (make sure adb is connected to the VM). It will appear as an application called `Student Message`. Locate the application on the VM and run it. If successful, what ever message you have typed should appear on screen. Test your program by writing multiple messages, one after another, and ensure that each appears after being sent. You may find the comments in `StudentMessageActivity.java` useful.

4.3 Attacking the malicious application (50 points)

For this section, you will look at the interface exposed by the malicious application uncovered in section 4.1. There exists a bug in the malicious application which allows any application on the same phone to make use of the malicious application's privilege. This means if we can trick a user into installing another application, even if the new application doesn't request any permission, we will be able to control the malicious application!

Your goal is to write a "jailbreak" application that performs the uninstallation of a package `jail.neo.lock` via a confused deputy attack. The skeleton of the application is in directory `unlockApp`. **Note:** You are only allowed to modify the unlock function in file `UnlockNeo.java`. Build and dispatch the application as described in the previous section.

Hint: Carefully read the Notepad tutorial and the supplementary material on inter-component communication, especially the intent examples.

Hint 2: To learn about uninstalling an application, Google `android pm uninstall`.

Hint 3: Due to the complexity of the phone, the attack might not always be successful, and sometimes it takes up to 1 minute to launch the attack.

Once the "jailbreak" application is launched, the text `It works! Now \*` should eventually appear on the screen, and you can also see: `java.lang.SecurityException: Neither user *** nor current process has android.permission. DELETE PACKAGES.` from logcat.

5 Issues and Workarounds

- I can't start the VM, VirtualBox complains about a non-existing vboxnet0 network
  - Go to your VM parameters->Network, first network card : choose an existing host-only network
  - If you don't have any host-only network, go to File->Parameters->Network, create one and then go back to your VM parameters to attach it
- At the home screen, the lock icon cannot be dragged to the right to unlock the VM

    - Press the escape key (click inside the VM to ensure the keyboard is being captured) and then try to drag the unlock icon.

- I can't move my mouse around inside the VM
    - Go to the VirtualBox Machine Menu and disable mouse integration
- `ant debug install` fails
    - Check through the messages and ensure that it is not a compilation error
    - Ensure that you have followed the instructions above and connected to the CURRENTLY RUNNING virtual machine. List the currently connected devices with `adb devices` and ensure that adb is connected.
- adb is unable to connect
    - This issue can be solved by killing the adb process and starting it again.
    - If this still does not work, ensure you are attempting to connect to the correct IP address.

6 Acknowledgements

This lab was prepared by Kevin Z Chen and modified for this semester by Ahir Reddy Thanks to Contagio Mobile [1] for providing the original sample.

[1] http://contagiominidump.blogspot.com/