Android Malware Exposed

An In-depth Look at the Evolution of Android Malware

Grayson Milbourne & Armando Orozco
August 2012
## Contents

Abstract .................................................................................................................. 3

Android Security Overview .................................................................................. 3

Android OS Overview ......................................................................................... 3

Android APK Overview ....................................................................................... 6

Overview of Tools Used for Static APK Analysis ................................................. 6

Overview of Tools Used for Dynamic APK Analysis ........................................... 7

SMS Trojans ......................................................................................................... 8

Server Side Polymorphosim ............................................................................... 9

Rootkits .............................................................................................................. 10

Spyware ............................................................................................................. 12

Drive-By Downloads .......................................................................................... 12

Android Malware Discovery Techniques ........................................................... 12

Future Predictions for Android Malware ............................................................. 13
Abstract

The world of malware targeting the Android OS is similar yet very different from malware affecting Windows. This presentation will explore the rapidly evolving world of Android malware and shed light on the various techniques used to exploit devices using this OS. We will start by looking at some of the earliest examples and compare them with the latest threats, exposing patterns and highlighting characteristics which are often unique to malicious apps. We will take an in-depth look at a wide array of Android malware ranging from Trojans to rootkits to man-in-the-browser attacks. We will discuss the tools used for analysis along with tips on how to organize the vast amount of data collected during the research process. Lastly we will talk about techniques to assist in the discovery of new malicious apps; a constant challenge with a rapidly growing and diverse app market space. This presentation will leave you with a clear understanding of how Android malware evolves, how to identify it, and what to expect in the year to come.

Android Malware Exposed
An In-depth Look at the Evolution of Android Malware

Android Security Overview

The dawn of the personal computer era gave birth to a new type of criminal, the hacker. These criminals, now collaborating and operating more efficiently than ever, aim to exploit any system containing interesting or valuable data; often with the goal of self-satisfaction and financial gain at the top of their minds. For the past few decades their focus has been on exploiting Microsoft Windows, largely due to the popularity of the operating system combined with the vast amount of personal data stored on PC’s. However today there is a new even more attractive target for hackers to exploit, the Android smartphone.

Smartphones have been growing rapidly in popularity over that past 5 years, and it is not surprising why. These devices put the internet, a camera, GPS tracking, and more into the palm of your hand. In 2008 Google released the first open source smartphone platform, Android. With its release, malware researchers everywhere knew it was only a matter of time before hackers started to exploit these devices.

Why Android smartphones?

The reason is simple; these devices contain more personal information than the average PC. When a smartphone app logs into your email account, it remembers the password so you never have to log in again. This behavior is common whether it is a personal email account, corporate VPN access, Paypal or Facebook. While this is convenient for most end users, it is even more appealing to hackers who are after this very information. In addition to apps keeping users logged in, smartphones also contain detailed records of your contacts and SMS history. This data is extremely useful to a hacker looking to commit identity theft.

In 2010 the first samples of malware were discovered. Since then researchers at Webroot have uncovered over 13,000 unique samples of Android malware.

Google is aware of security flaws in their OS and have rapidly released new versions to address these flaws. Unfortunately, 85% of Android users are still using 2.x versions with many known vulnerabilities. The problem is that most device manufactures and carriers do not plan to update older devices to the newest and most secure version of Android. The reason is mainly due to profits. Carriers and manufactures would much rather users shell out a few hundred dollars for the newest device. Google has largely addressed this issue with version 4.x, also known as Ice Cream Sandwich or ICS. ICS aims to create a new standard where all devices are capable of upgrading as soon as a new release is made available.
Android OS Overview

How does Android work?
Android is built on a Linux kernel with a few additional drivers to support power management and the various features often found on smartphones, things like WiFi, GPS and a camera. These kernel drivers lay the foundation for Android's first primary security feature, a virtualized runtime environment known as Dalvik. The Android OS is architected such that every new application launches in its own unique instance of Dalvik. This is a really nice security feature because it prevents applications from accessing or being aware of other applications which are also executing on a device. This technique creates a secure execution environment and prevents data leakage between applications. The Dalvik VM interacts with a number of core libraries which lend functionality to the application running within. But how is the access to various functionalities controlled? This brings us to the next major security element of the Android OS, the permissions model.
Permissions are requested by an application during the install process to grant access to various features and functionalities on a device. Currently there are 124 unique permissions which are categorized into 11 top level groups. These permissions range from services that cost you money to access of your personal information. A full list of these permissions can be found here. The permission model is a good security feature because it forces developers to explicitly ask for the specific functionalities needed for their application to function. These permissions are displayed before any application is installed and can also be viewed post installation. The downfall is that users cannot be expected to understand all 124 permissions or the associated risks with a few specific permissions. Not having the comprehensive knowledge of the app the developer will have, it is also impossible for users to know which permissions are actually needed by an application. We will come back to this point later when we examine malware which exploits this very conundrum. One last gotcha of the permission model is that it is all or nothing. When installing an application, the permissions needed are displayed; but users do not have the ability to decline certain permissions thereby choosing which functionality an application can access versus which are forbidden. With these nice security features, why is it that Android has a malware problem but iOS does not? To better understand this we need to fully understand the application (APK) format which is unique to Android devices.

### Android APK Overview

Android application installers, or APK's, are unencrypted archives which contain all the necessary files for a functioning application. Extracting the contents is as easy as opening an APK with WinZip or similar archive software. The APK archive contains a few different folders and files. The first of the folders is the Meta-INF directory which contains the Manifest.mf, Cert.rsa and Cert.sf files. The Manifest.mf and Cert.sf files contain a list of resource paths along with the associated SHA1-Digests hashes. The Cert.rsa file contains the application certificate which is required for every Android application. The next folder is the Res (resource) directory which contains the files pointed to by the Manifest.mf and Cert.sf files. These are typically .xml and .png files. Some APK's might contain an Assets folder which is just another directory to store files needed by the application. This brings us to the final three files which are always stored at the root of the APK structure. These are the Resources.arsc, AndroidManifest.xml and Classes.dex files. Resources.arsc contains a list of the resources used by the APK. The AndroidManifest.xml file contains detailed information about the program. This includes the name, version, and permissions used by the APK. Despite the file having a .xml extension, this file is not a true xml file, but rather an Android binary which is best viewed using Apktool. The same can be said for the Resources.arsc file. The final file, Classes.dex, contains the compiled source code for the application and functions as the Dalvik executable. What makes Android so appealing to hackers is the ease at which an application can be disassembled, analyzed, changed and recompiled. We will now cover the tools used for APK analysis.

### Overview of Tools Used for Static APK Analysis

When considering binary analysis there are two approaches you can take; static and dynamic. We will cover tools you can use for both approaches. When researching an application, whether it be Windows, Android or iOS, you will want to have a peek inside the binary. Android applications are primarily written in Java and getting to the source or byte code is very easy with the assistance of a few tools. Static analysis can yield a lot of useful information about an APK such as permissions requested, which API’s are called and specific hardware requirements. While there are various ways of doing this, the best way is using a disassembler which can make sense of the machine code within the compiled executable. For Android there are a few tools which are publicly available. We will cover a couple; one that can disassemble the Dalvik executable and another that can decompile it into readable Java. A great tool for disassembling a .dex file is Apktool. It takes the Classes.dex file and disassembles it into Small Android assembly language. Apktool can decode the AndroidManifest.xml and Resources.arsc files. It also has options to debug Smali and reassemble or build an APK. With the output from Apktool you are able to review the AndroidManifest.xml, view payloads, and peer into the application's functionality. Dex2Jar is a similar tool but the output is a JAR file instead of Smali. Java is easier to read and makes more sense to most researchers. The output will not mirror the actual source code, rather produce a solid representation of it which is useful for understanding how an application functions. Once you have a JAR file you can view it in a Java disassembler such as JD-Gui.
Overview of Tools Used for Dynamic APK Analysis

Dynamic analysis can also yield positive results by capturing runtime data such as system events and network traffic. The Android SDK, which is part of the Android development environment, comes with some very useful tools. It provides access to all of the latest OS versions, an emulator and a debugger. You can build emulators for the various different OS versions as well as devices with different hardware specs. Having an emulator allows you to install Android applications you built or came across during the research process. The other great tool available with the Android SDK is the Dalvik Debugger. It lets you view events using LogCat, Android's debugging output viewer. As you click through an application’s interface, LogCat will capture these events along with any system events. It also allows you to emulate SMS, phone calls and geographical location data. There are other dynamic analysis tools such as DroidBox and TraintDroid which are custom ROM’s built to capture events by applications. They are designed to track Network traffic, SMS, phone activity, attempts to access the address book, geographical location and other data on a device. Now that we have covered analysis tools, it is time to look at the various examples of malware that have been discovered on Android.
SMS Trojans

Android malware has evolved its tactics and distribution over the last two years. Two big news makers for Android malware were TrojanSMS, a premium-SMS Trojan, and DroidKungFu, a bot with rooting capabilities. We will look at how these two have evolved in delivery and tactics.

The premium-SMS Trojan is a lucrative form of malware that is simple to develop and has the added benefit that it uses alluring tactics which users tend to fall for. Recently, two groups that were caught distributing SMS Trojan's received some justice. The hackers responsible for the Foncy campaign were arrested in France with damages estimated around $150,000 and the other was a company, A1 Agregator Limited, who was responsible for the payment system RuFraud. They were fined $78,300. The SMS Trojan 'FakePlayer' was the first fake app to charge for its use. It poses as a legit media player but would send out premium-SMS messages without the user's knowledge. There was little sophistication to the malware other than tricking users in to running the program. These APK's are very small in size, around 15KB, and when launched display a message in Russian, “Click OK to access the video library” and a second message of “Wait, requested access to the video library.” While you are waiting, premium-SMS messages are sent, costing the unsuspecting user money. Below are snapshots of the icons and a code snippet which sends the SMS message.

```
localSmsManager.sendTextMessage("5555", null, "798657", null, null);
```

Although relatively new as it is, the SMS Trojan has evolved throughout the last year. What we are primarily seeing now are fake installers. These are apps that claim to be installers for a legitimate app, often found free if searched for on the Google Play market, but come with a price. Once a user installs one such app they will see a 'License Agreement' or 'Terms of Service' which states they must send three, fee based SMS messages to receive the app. Rates vary depending on the country and carrier. Typically these three messages will go to different numbers, each charging a different fee. The screenshots below show examples of the screen when your first run the app and the terms you must agree to.

Example rates using the premium-SMS numbers in the screenshot would be:
- # 7151 range of 33.87-40.00 rubles US $1.10-1.30
- # 9151 range of 101.60-140.42 rubles US $3.30-4.56
- # 2855 range of 170-203.20 rubles US $5.52-6.60

Total cost 137.17-383.62 rubles US $9.92-12.46

Most often users do not ever receive the app they thought they paid for.
Server Side Polymorphism

The authors of these Trojans have made some strides in their delivery method techniques. Let us take a look at how they have brought polymorphism to Android malware and rogue Android markets. The package name of an Android app is one way to identify an APK. In an attempt of evading detection, hackers have implemented ways of compiling new apps with generic and/or random package names such as, ‘vnpyso.wbkwyv’ or ‘noogeil.hphah. aegaeg’. These package names have been seen to change a couple of times a week and as often as each download attempt. Another tactic for evasion is code and resources modification. Making minor changes to these files in an APK changes the binary hash which makes it undetectable if a security vendor is detecting based on MD5 hashes. In Figure 1 and Figure 2 we show two examples of this technique. These are samples taken from the same download link found on an alternative Android market. Notice the variances in the Java Classes and the Send Method between the two downloads.

![Figure 1](image1)

![Figure 2](image2)

It is fair to say that these Trojans must be very lucrative because we have found dozens of rogue Android markets which only serve up premium-SMS Trojans. Based on their appearance and the amount of detail included in them, a good deal of time has been put into these sites to make them appear legit. They have many elements that a legit Android market would have including; device reviews, forums and mobile related news. Below is snapshot of one offering a free copy of Angry Birds Rio.

![Screenshot of Angry Birds Rio](image3)
Rootkits

We have encountered numerous examples of Trojans with root capabilities. These Trojans often have command & control functionality similar to what has been seen with PC botnets. Because these apps root, they gain escalated privileges and are able to bypass Android’s permission model; thereby granting access to all functionality on the device without user notification. Taking advantage of known exploits in the Android OS, malware authors bundle these exploits in their APK’s. The rooting exploits are the same ones made available by hackers for those willing to intentionally root their device. The two most prevalent ones target versions 2.1 and 2.2 of the Android OS, rage-againstthecage and exploid. Two notable examples of rooting malware are DroidDream and DroidKungFu. At the time these Trojans were discovered, version 2.1 and 2.2 were the most distributed versions of the Android platform. At the time of writing this paper, these versions are still on 26% of Android devices, leaving a large number of users exposed to these exploits. Below is a code snippet from some of DroidDream’s rooting functionality.

DroidKungFu variants are known to hide rooting tools by changing the name of the exploit files such as ratc, secbino, and webview.db.init.

Keeping with the same theme of using existing exploits, new OS versions brought new exploits and targets for malware authors. GingerBreak was one of the first rooting apps available for v2.3 of the Android OS. Soon after its release, malware authors were bundling it with their apps. GingerMaster was the first of these and is a cousin of DroidKungFu having the same functionality with a GingerBreak twist. The exploit comes as an ELF file disguised as an image file. Figure 3 shows a snippet from the ELF file.
As you can see malware authors typically do not reinvent the wheel and will reuse code and use publically available exploits. We are all fans of the people who put time in to find and utilize these exploits but the bad guys appreciate them, too.
Spyware

Other types of threats are those that spy on you or steal your data. There are a number of apps that are the equivalent to commercial keyloggers found on PCs. These apps offer their services to 'track' your kids, spouse or employees. These behaviors are easy to incorporate into an app and this begins with the easy task of requesting the necessary permissions. For example, requesting ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, and READ_SMS will grant you access to SMS messages and GPS location. Of course you will have to add the appropriate code, and if it is not a rooted device, permissions must be approved. Threats which have used these spying techniques are NickySpy, Spitmo, GGTracker and GoldenEagle. NickySpy is interesting in that it utilizes the MediaRecorder() class to turn on the microphone and discretely record and save conversations to the SDCard. It is also able to send captured data to a remote server, although this functionality is not hard wired in. Below is a snippet of the function responsible for voice recording.

```
public void callrecord()
{
    try
    {
        MediaRecorder localMediaRecorder = new MediaRecorder();
        this.recorder = localMediaRecorder;
    }

    public void onCreate()
    {
        super.onCreate();
        PowerManager.WakeLock localWakeLock =
                ((PowerManager)getApplicationContext().getSystemService("power")).newWakeLock(1, "RecordService");
        String str = localSimpleDateFormat.format(localLong);
        this.filename = str;
        stopCallRec();
        callrecord();
    }
```

Drive-By Downloads

In May of 2012 the first reports of drive-by downloads targeting Android browsers were seen. Drive-by downloads have been the bane of Internet browsing in the desktop environment for many years now and this infection vector has evolved to target Android devices. This is just another example of how fast the Android malware landscape is evolving. This particular threat typically utilizes a hidden iframe tag located at the bottom of a hacked website. These websites specifically look for Android user-agent strings before serving the malicious iframe, therefore the payload would only be delivered when visiting the site with an Android browser. When the site is visited with an Android browser, the iframe would trigger the browser to download and execute the payload. Another common vector for infection which primarily targets PCs is the web exploit kit. BlackHole is a very common example which cycles through various PC based exploits in its attempt to execute a malicious binary. It is not long before web exploit kits will have a module looking to infect mobile devices.

Android Malware Discovery Techniques

Identifying Android malware can be a challenging task for researchers. Many times, identifying an app as malicious based on high level data can be misleading and lead to complications in the research process if the researcher is not positive of what they are looking for. The permission model is a good place to start; but because an app requests a certain permission does not make it malicious. Often times, we must dig deeper to find out the true intent of the app. Digging deeper can be automated by utilizing static and dynamic tools. You can group similar apps based on permission’s, APIs and other data found in the Android-Manifest.xml. Being able to identify common code quickly can help save time and allow for better grouping. We often see malware authors reusing the same code but trying to hide it by changing package names and class paths. This causes topical data to look different but at a lower level the functionality will be similar.
With Android's built in security features, malware is limited in what it can do. Identifying similar code with a heuristic approach can give you a better grasp on recognizing malicious behaviors. There are various techniques for this ranging from code normalization to function and class classification. To successfully discover new threats, a combination of all discovery techniques combined with ample manual research is a must.

Future Predictions for Android Malware

In the coming months and years we expect Android malware to continue to grow as the Android platform has seen amazing growth in the mobile market space. What types of trends and attack vectors will we see? For one, we can expect many more premium-SMS Trojans as these are big money makers. Trojans will continue to be bundled in repackaged APK's and disguised as legitimate applications. With 900,000 daily Android activations worldwide, social-engineering tactics will continue to be used to trick users into installing malware. There have only been a handful of Android banking Trojans seen thus far and we expect that number to grow. These infections generate a significant source of revenue in the PC space and would be expected to do the same in a mobile device world. The hackers behind Zeus and SpyEye will not pass up the opportunity to exploit mobile devices. Recently we have seen Ad pop-ups in mobile browsers which is a great attack vector for Fake AV apps to be distributed. It is easy to trick a user into thinking they are infected with malware, suggesting they install an app, and then extort the user with a pay-for-play scheme, steal data or both. We expect to see more and more drive-by download attacks which will utilize Ad delivery, SMS spam and browser exploits to install payloads on unsuspecting users' devices. As long as people continue to put their trust into their handheld devices and use them as a replacement to their PC's, hackers will continue to discover exploits and new techniques to gain access to the wealth of information stored on these devices. The game of cat and mouse is not going anywhere any time soon.