Android Malware Forensics: What It Does and How to Find It

by
Corrie Erk
corrie.erk@gmail.com
May 2015
ABSTRACT

Cyber threats have in recent years expanded to target the technology that continues to be released and developed. Now, desktop devices are not the only threat for malware, but mobile phones are the new, up-coming target. Mobile forensics in general is still in its infancy when it comes to acquisitions and analysis, as is reverse-engineering the malware targeting these devices. With Android devices holding the majority market of mobile users, the most mobile malware being created (while not very sophisticated) targets these devices specifically. After testing two samples of Android malware, patterns in Android malware behavior are developed, as well as how to forensically analyze compromised devices. It is found that rooting an Android device and taking a physical image of the device will provide the most information related to a compromise and the malware itself. The common folders related to applications can be analyzed to find malware artifacts; the data folder, the downloads folder, the app and app-lib folders, and the dalvik-cache folder. Evidence of malware can be found in these locations, and suspicious files can be extracted and reverse-engineered to read the raw code of the malware to have a full understanding of what its functions are.
# Table of Contents

**Abstract** .......................................................................................................................... 2  

**Introduction** ....................................................................................................................... 5  
  Proposal ............................................................................................................................... 6  

**Background** ....................................................................................................................... 7  
  Android Mobile Devices ......................................................................................................... 7  
  Partitions ............................................................................................................................... 7  
  Application Architecture ......................................................................................................... 8  
  Android Forensics .................................................................................................................. 9  
  Cyber Threats ....................................................................................................................... 10  
  Android Malware .................................................................................................................. 11  
  Malware Analysis ................................................................................................................. 13  

**Methodologies** ................................................................................................................... 14  
  Mobile Phone ....................................................................................................................... 14  
  Forensic Machine(s) ............................................................................................................. 15  
  Procedures ............................................................................................................................. 15  
  Forensic Collection .............................................................................................................. 19  

**Findings** ................................................................................................................................ 22  
  Malware Samples .................................................................................................................. 22  
  Godwon .................................................................................................................................. 22  
  Social Path ............................................................................................................................. 23  
  Static Analysis ....................................................................................................................... 23  
  Godwon .................................................................................................................................. 24  
  Social Path ............................................................................................................................. 26  
  Behavioral Artifacts ............................................................................................................. 29  
  Godwon .................................................................................................................................. 29  
  Social Path ............................................................................................................................. 30  
  Forensic Artifacts .................................................................................................................. 32  
  Downloads Folder ................................................................................................................... 32  
  App Folder ............................................................................................................................. 33  
  Data Folder ............................................................................................................................. 34  
  App-lib Folder ......................................................................................................................... 36  
  Dalvik-cache Folder ............................................................................................................... 37  

**Conclusion** .......................................................................................................................... 38  
  Behavioral Patterns ............................................................................................................... 39  
  Forensic Patterns .................................................................................................................. 40  
  Triage Recommendations ...................................................................................................... 41  
  Future Research ..................................................................................................................... 41  

**Works Cited** ....................................................................................................................... 44
INTRODUCTION

The use of technology in the 21st Century has evolved exponentially, providing tools for people to learn, work, connect with one another, and entertain in a convenient and efficient manner. With this growing technology becoming a mainstream part of daily life, people rely more heavily on the use of their mobile devices, which incorporates a way for users to accomplish all of their daily tasks. In the cyber world, the heavy reliance on mobile devices by a large population causes the amount of malware being created and targeting these platforms to increase. A mobile threat report released by the security group Lookout revealed that the number of Android mobile phone users who became infected with malware increased by 75% between 2013 and 2014 (Lookout, 2015). These types of malware function as spying devices, and include ways to steal personal information or commit fraud. In June of 2014, a mobile malware family called WireLurker surfaced in the cyber world, which created scares across the security industry because of its sophistication in infecting cross-platforms (Mac OS X and iPhone) as well as being discreet in that it does not require the mobile devices to be Jailbroken to gain administrative access with reduced security measures (Palo Alto Networks, 2014).

One large issue surrounding mobile malware threats in the forensics industry is trying to identify what the malware is and what it has done with the device. Was it used as a listening device? Was it used to commit a cyber crime? Were data and other private information exfiltrated? These are all questions that are asked during a malware infection of a normal desktop device, and those questions remain the same for compromised
mobile devices as well. At this time, the forensics community is at a benefit in that Android malware and other mobile threat types are not nearly as sophisticated as they are for Windows platforms. The command and control mechanisms used often do not function as expected, and the configurations are not flexible and unpredictable. Additionally, the malware will take limited steps to concealing itself, therefore leaving identification of the infection as a relatively easy task (Peters, 2014).

On a Windows device, malware saves its payloads and configurations to common places on the device, such as the user’s AppData folder, or the Windows Temp folder. These are popular places to look when first attempting to identify a compromise. Patterns such as these are not currently documented, as the research around mobile malware families are still in its infancy. Further research needs to be performed on mobile malware in order to create a set of standards for how forensic analysts can examine these crimes. Additionally, patterns need to be discovered to better analyze infected devices as well.

**PROPOSAL**

This project will serve to provide information about Android mobile device malware, as well as what to look for when forensically investigating a potentially infected device. Malware forensic techniques and artifacts for the Android operating system will result from research and testing performed.
BACKGROUND

ANDROID MOBILE DEVICES

The Android mobile operating system is a platform acquired by Google in 2005 when the company was just a startup (Elgin, 2005). In the past ten years, the platform has become the most commonly used mobile operating system, being supported not just on smartphones, but tablet computers, televisions, cars, and even wrist watches. In addition to making phone calls and send or receive text messages as a normal cellular device might, Androids use applications to add a variety of functionalities and purposes to the smartphone. The operating system is open source, meaning that the source code is available to developers who can create modifications and applications that allow users to utilize the device in a way that’s best fit for them.

PARTITIONS

Android devices contain multiple partitions to store data. The internal partitions consist of boot, system, recovery, data, cache, and misc. (Raja, 2011).

- /boot: Contains the phone’s kernel and ramdisk required to make the phone boot.
- /system: Contains the operating system, including the user interface and system applications that are pre-installed on the mobile phone.
- /recovery: Stores a second boot option used for maintenance and recovery options.
• /data: Also known as the USERDATA partition, contains the data related to the user, including contacts, messages, settings, applications, etc.

• /cache: Contains frequently accessed data and application components

• /misc: Contains miscellaneous settings to be turned on or off, such as hardware settings.

An external partition, /sdcard, may also be present representing a SD card often used as extra storage space for applications, media, documents, etc.

APPLICATION ARCHITECTURE

Android applications are developed in Java, and then are compiled into an application package file commonly known as an APK file (the file extension being .apk). An APK file is similar to a zip archive, containing a directory of files required by the application. This directory of files can be viewed by adding a “.zip” to the end of the file name, and using a zip file program to extract the files.

Once unzipped, the APK contents will include the AndroidManifest.xml, which is required to be located in the root directory of every application. This XML file contains app information for the operating system, such as package names for Java, application components and which processes host those components, permissions, versions, libraries, etc. (Android Developers, 2015). The zipped APK may also include application certificates, libraries of compiled code, application resources and assets, and java-compiled executable files (.dex files).
Applications (more specifically, .dex files) are executed in their own Dalvik Virtual Machine (DVM) to run the Dex bytecode (translated from Java bytecode) of the application (Hildenbrand, 2012). Having each application contained to their own DVM allows for multiple applications to be run at the same time, and restricts what each application can access on the rest of the system with or without permissions from the user. (Barrett, 2014).

**ANDROID FORENSICS**

Mobile forensics in general is a fairly new specialty in the digital forensics field. Although, between the large mobile phone operating systems used today (Android, iOS, Windows, and Blackberry), Android may be the operating system that is easiest to acquire and analyze in the field due to its open source nature. The mobile company Cellebrite\(^1\) provides forensic solutions to enable analysts to acquire, parse, and analyze data on most cellphones and smartphones available in the market. Cellebrite’s UFED software allows for three types of acquisition for Android devices; Logical, File System, and Physical.

\(^1\) [http://www.cellebrite.com/](http://www.cellebrite.com/)
Logical acquisitions will collect the least amount of data, but will include SMS data, contacts, call logs, and media on the device. File System collections include the data from a logical acquisition, but also can collect standard files, including those which are hidden. Lastly, physical acquisitions collect all of the above information types, but also include deleted data.

In order to access the full data on a device, Androids need to be rooted. Rooting a device gives the user full privileges and control on the device (root access) with access to any file or folder on the device. Without root access, many files and partitions are hidden from acquisitions and analysis. Therefore, it is common to see analysts root Android devices before acquiring them, in order to ensure the most data is captured and accessible.

CYBER THREATS

Fifteen years ago, the integration of technology in the daily lives of the population was not nearly as prevalent as it is now. Over three billion people in the world have an Internet connection, compared to fewer than half a million users back in 2000 (Internet Live Stats, 2015). This means that approximately 40% of the world’s population is susceptible to cyber attacks. These attacks can often include the exploitation of vulnerable network and technology infrastructures, malicious emails social engineering recipients into revealing personal information such as passwords or credit card numbers (a method known as phishing), sniffing network traffic to collect information transferred through the Internet, and tricking users into installing malware on a system. These types
of attacks happen every day, as displayed by a cyber map put together by the security company, Norse, found at http://map.ipviking.com/.

In 2014, the computer security company Sophos released a Security Threat Trends report predicting the cybersecurity concerns of 2015. The predictions include more flaws in widely-used software that have gone undetected in previous years, new rootkits and bots infecting devices using different methods than in the past, and an increase in attacks against mobile platforms. Of the mobile threats, a focus on exploit kits targeting mobile payment systems to collect credit and debit card information is expected to increase (Lyne, 2014). The predicted focus on mobile platforms is caused by the increasing growth in the use of smartphones as a tool to pay for products and services, store personal information, and communicate such personal information to one another via email or text.

Android Malware

When the smartphone market was analyzed at the conclusion of 2014, it was found by the International Data Corporation (IDC) that the Android operating system held 81.5% of the smartphone market share (IDC, 2015). In addition, Forbes released an article stating that 97% of mobile malware can be found targeting Android devices (Kelly, 2014). This is not a coincidence. Malware creators want to target the largest population of users as possible when creating an attack for it to be the most effective, which is why Windows devices have more susceptible to malware than Mac devices in the past. In the smartphone world, Android’s open source architecture and majority users allow for the development of cyber attacks to these devices to be the most beneficial.
In addition to the ease in development and the high market, there are limited requirements for uploading an application to the Google Play Store. According to the support page on Google’s website, a user only has to go to their own Google Play Developer Console, click “Add new application,” select a name and language preference for the app, and upload the APK. The only security requirement is for the application to have debugging disabled before publishing in the store (Google Play, 2015). Therefore, it is relatively simple for a user to upload and publish malicious applications. Although, Google has recently begun scanning applications as they are uploaded to Google Play in order to detect and remove malicious instances found in the store. Users also have the ability to disable app-specific security controls, which would prevent a user from installing applications that are not sourced from the Google Play Store (such as external website downloads).

There has been a noticeable increase in malware found in the Google Play Store in recent years. Between 2011 and 2013, the number of mobile applications infected with malware discovered in the store nearly grew 400% (Miners, 2014). Most of the malware detected both internal and external to the Store are used to advertise or promote other products and services, and collect information stored on the phones (device IDs, contacts, text messages, GPS coordinates, email addresses, phone conversations, credit card numbers, etc.) which can later be sold to third parties or used against the victim’s will.

Although, analysis of Android-specific malware thus far has shown that the infections do not always take immediate action, or take action at all. Most malware discovered remains dormant for a period of time, sometimes as long as a month, before
any symptoms are recognizable to the user. A recent infected app, called Durak, was a card game application, which waited 30 days before pushing threatening pop-up advertisements, scaring the user into thinking they needed to perform a security update or fix to their phone, when ultimately they were led to other malicious websites to download more malware, or asked to provide credit card information to remediate (Olivarez-Giles, 2015). Additionally, Android malware has been relatively immaturely created and lack sophistication compared to their Windows counterparts. The command and control mechanisms hardcoded into the applications often do not work properly which defeats many purposes and functionality of the malware. The samples studied thus far also do not show any methods to conceal itself while installed (Peters, 2014).

MALWARE ANALYSIS

In the information security world, having the skills or a service to analyze malware has become a must in many companies. The purpose of reverse-engineering malware is to have an understanding of what the malware does. What is its functionality? What is the threat? What can be done to prevent it? How can it be detected outside of Anti-virus? What needs to be done to remediate the threat? There are several automated services available for purchase and for free which will analyze malware to help answer these questions, or those with enough resources can build a sandbox environment in order to answer these questions themselves.

Behavioral analysis of malware is a process of scrutinizing the malware to determine what it is meant to do. One way this can be done is through static analysis, where the code is reverse-engineered to understand what it was designed to do without
execution. A second method is to perform dynamic analysis, where the sample is executed in a secure environment and monitored to collect information of how it behaves in-action (Erk, 2014). These practices are commonly applied when investigating malware infections on desktop devices (such as Windows or Mac operating systems) but the practices can be applied to other platforms as well with variations in tools used to perform the analysis.

**METHODOLOGIES**

**MOBILE PHONE**

The mobile phone used to perform this research is a Samsung Galaxy S4 running Android Kit Kat version 4.4.2. The phone is configured to disable the built-in security settings, which would prevent malicious files from being downloaded and installed on the phone. Additionally, the phone is rooted using an application called Towelroot. Towelroot allows a user to root an Android device with the press of a button. Disabling security features on the phone allows for the malware to easily be installed on the Samsung device, while rooting the mobile phone allows access to the system’s full contents, which are required for imaging and analysis.

---

2 https://towelroot.com
FORENSIC MACHINE(S)

The forensic machine used for acquisition of the phone and analysis is a desktop device running Windows 7 Professional. Installed on the machine is a suite of tools by Cellebrite, including UFED4PC for imaging, and UFED Physical Analyzer 4 for analysis.

The handling of malware was performed on a Macbook Air running OS X Mavericks. Android File Transfer for Mac is installed on the device to aid in moving the Android malware samples to the Samsung device to be installed.

PROCEDURES

1. Perform a factory reset on Android device
   a. Power down the device
   b. Hold the volume up button, power button, and home button simultaneously until the Samsung logo appears. Then release.
   c. When the Android System Recovery screen appears, select “Wipe Data/Factory reset”
   d. Select “Yes – delete all user data” to confirm wipe
   e. Device will restart
2. Setup device and disable security features

   a. Go to Settings > Security and ensure “Unknown Sources” is checked to allow apps to be installed outside of the Play Store and uncheck “Verify Apps” to prevent the blocking of potentially malicious applications.

   ![Security Settings](image)

   **FIGURE 2: SECURITY SETTINGS**

   b. Enable developer options by going to Settings > About Device and tapping “Build Number” repeatedly 7 times.

   c. Go into Developer Options under Settings, enable “USB Debugging” and ensure “Verify apps via USB” is disabled.
3. Root the phone to ensure the most access to the device’s system is accessible to image in the next step

   a. Using a mobile browser, download Towelroot\(^3\) to the phone to easily root the device. The file “tr.apk” will be saved to the “Downloaded” folder.

   b. Run the tr.apk to install the app

   c. Once installed, open the app and click the “make it ra1n” button to root the device

---

\(^3\) [https://towelroot.com](https://towelroot.com)
d. Ensure the device is rooted by installing and running a root checker app from the Google Play store, such as Root Checker Basic.

![Root Checker Basic](image)

**FIGURE 5: ROOT CHECKER BASIC**

4. Take a clean physical image of the mobile phone using Cellebrite UFED4PC (see Forensic Collection section below for details).

5. Infect phone with malware

   a. Install a tool on the forensic desktop to easily transfer files from the desktop to the Android device.

      i. Android File Transfer for Mac used to save the malware file to the Download folder on the Android

      ![Download Folder](image)

      **FIGURE 6: DOWNLOAD FOLDER**

   b. Run and install the malicious APK, accepting all security permissions.
   a. NOTE: Since Android malware can wait up to 30 days or more to show any symptoms or functionality, this would be an ideal timeframe for letting the sample run before analyzing. Although, it is understood that this is not always a reasonable timeframe to perform analysis, therefore sample was left to run for approximately 5-8 hours.
   b. Document any running processes or noticeable changes to the device (such as installed applications, services, symptoms, etc.)

7. Take a second, infected physical image of the device using Cellebrite UFED4PC software.

8. Perform analysis of infected image (in comparison to clean image) using Cellebrite Physical Analyzer software.

9. Repeat steps 1-8 for each sample analyzed.

**FORENSIC COLLECTION**

A physical image of an Android phone can be done using the Cellebrite UFED 4PC software as long as USB Debugging is enabled on the smartphone in order to acquire data. Cellebrite’s mobile kit comes with a series of USB cables that can be used to attach the phone to the forensic machine, or the original cable can be used. The
following procedures can be followed in order to take a physical forensic image of an Android device.

1. Launch UFED 4PC and at the main screen, select “Extract from Mobile Device”

![FIGURE 7: UFED 4PC - 1](image)

2. Plug in the Samsung device for UFED 4PC to detect it automatically, or browse through the make/models listed in the tool to select the device to be acquired
3. Once the device is detected, select “Physical Extraction”

4. At the next screen, select “ADB” as the mode for acquisition (the recommended Boot Loader option errored out periodically during acquisition when tested)

5. Select the location to save the image by clicking “Change Target Path”

6. Disconnect and reconnect the device from the machine. Click “Continue” to proceed to acquisition.
7. The acquisition will start, and prompt a screen when complete.

**FINDINGS**

**MALWARE SAMPLES**

Both malware samples used can be downloaded from the Contagio Mobile Malware Blog. The two samples to be tested are commonly named as Godwon and Social Path malware.

**GODWON**

The first sample to be tested is known belonging to the Godwon family, which is an Android information stealer malware. According to Trend Micro, the malware is a tool for sextortion groups in the Far East to gain contact information to make further threats (Flores, et al., 2015).

4 http://contagiominidump.blogspot.com/
The sample tested has the following properties:

- File Name: godwon_0CCF75E179D91CCBD86722014F014607.apk
- MD5: 0CCF75E179D91CCBD86722014F014607
- File Size: 22435 bytes

**SOCIAL PATH**

The second sample tested is known as Social Path malware. This malware, called Save Me, is advertised as a tool to help protect a user’s privacy by alerting when a user’s photo was uploaded or posted somewhere on the internet, but will steal data from the phone once installed instead. Samples were at one point found on the Google Play Store, as well as being distributed through spam campaigns and popular social media websites such as Twitter or WhatsApp (Linden, 2015).

The sample tested has the following properties:

- File Name: save me_78835947CCA21BA42110A4F206A7A486.apk
- MD5: 78835947CCA21BA42110A4F206A7A486
- File Size: 2318602 bytes

**STATIC ANALYSIS**

Extracting the APK archive to reveal its contents will reveal a file called “classes.dex” with each application. This file holds the classes the application uses in a dex file format to be understood by the Dalvik Virtual Machine. Using a tool such as
dex2jar\(^5\) for Windows will convert the .dex files to a zipped jar file saved to the directory of the dex2jar folder. This can be done first by extracting the zip file for dex2jar to a location on the machine, and copying the classes.dex file from the malicious APK within the dex2jar folder. Running the command “d2j-dex2jar.bat <source classes.dex>” will convert the .dex file to a separate .jar file. A java decompiler, such as JD-GUI\(^6\), can then be used to open and parse the jar file to see the contents of what the APK was programmed to do.

![FIGURE 10: DEX2JAR COMMAND](image)

**GODWON**

After converting the classes.dex file for the Godwon sample to jar, the file was opened using JD-GUI to analyze the class files. Within the directory tree, there is a service called “GogleService” being used by the malware. Analyzing the code for this class, it appears as though the malware is engineered to collect the phone’s contact

\(^{5}\) https://github.com/pxb1988/dex2jar

\(^{6}\) http://jd.benow.ca/
information and Skype account information, then exfiltrate that information to hxxp://118.193.211.38/saves.ashx.

The “MainActivity.class” file found within the Dex file is programmed to start the GogleService class.

FIGURE 11: GODWON GOGLESERVICE

FIGURE 12: GODWON MAINACTIVITY
Overall, the malware is meant to start the GogleService class, which is used to steal contact and Skype information to send to cybercriminals in China, according to the geolocation of the 118.193.211.38 IP address data is exfiltrated to.

**SOCIAL PATH**

Using JD-GUI to open the jar file containing the converted Dex contents, many class files are observed.

**FIGURE 13: SOCIAL PATH CLASS FILES**
Within the “savemebeta” folder, the class files original to the application are stored. Parsing through the information shows evidence of the collection of contact information and SMS data, call logs, a table being created to store usernames and passwords, updates, etc. All this information is sent to a site, topemarketing.com, once collected.

The following sites are accessed and coded to post data to:

Script coded within the application identify a page that the user is lead to that prompts him or her to sign up for the service and fill out a form (which would contain personal or private information to protect their data) to be stored in a database.

```java

} else if (data.length == 0) {
    Toast.makeText(this, "Please fill out all fields", 0).show();
    return;
}
```

**FIGURE 15: SOCIAL PATH "FILL OUT ALL FIELDS"

Once data is saved within the application, the user will see confirmation that their data was saved within the database.

```java

localCursor.close();
Toast.makeText(this, "Your SMS Saved Successfully In Our Database", 1).show();
Toast.makeText(this, "Go to our website to get your sms", 1).show();
return null;
localUnsupportedEncodingException = localUnsupportedEncodingException;
localUnsupportedEncodingException.printStackTrace();
continue;
paramVarArgs = paramVarArgs;
paramVarArgs.printStackTrace();
```

**FIGURE 16: SOCIAL PATH "SMS SAVED"

Overall, the application appears to trick the user into thinking it will protect data which is put into it, when in reality the data is saved to a database on the phone which gets exfiltrated to a remote server behind topemarketing.com.
BEHAVIORAL ARTIFACTS

The malware samples were placed onto the test Android device using the Android File Manager installed on a Macbook Air. The files were transferred and saved to the Downloads folder on the Android. The malicious APK files were executed and installed on the rooted Samsung device from that location.

GODWON

When installing the Godwon sample, the Android system will prompt the user asking to accept the installation with the privacy and device access the application is requesting. Accepting the privacy and device access requests and selecting “Install” verifies the application is installed. It then can be seen as an application in the main menu of the device, written in Korean.

FIGURE 17: GODWON INSTALL
Looking at the active services and running processes on the device reveals evidence of the application running and the GogleService running.

![FIGURE 18: GODWON APPLICATION](image1.png) ![FIGURE 19: GOGLE SERVICE](image2.png)

No other noticeable symptoms were observed other than a newly installed application to the home screen and evidence of it running.

**Social Path**

When installing the Social Path malware sample, the Android will prompt asking to grant the app permissions to perform a long list of functions including the following:

- Directly call phone numbers
- Read phone status and identity
- Edit/Read/Send/Receive text messages
• Modify and Read contacts
• Write call log
• Full network access
• View network/WiFi connections
• Access and pair with Bluetooth devices
• Retrieve running apps
• Draw over/Close apps
• Install shortcuts
• Modify system settin

Once the application is installed, launching it will show a splash screen of the app’s logo and lead to a Terms of Service page, with a link to the topemarketing.org site. Proceeding past the terms of service will show an information screen written in French.
At this point, the Google Service crashed, therefore crashing the application. Also at this time, the site http://topemarketing.com and http://topemarketing.org were down.

**FORENSIC ARTIFACTS**

All findings produced were discovered using Cellebrite’s UFED Physical Analyzer v 4.1.0.178 after running the built-in Android plug-ins to parse the data.

**DOWNLOADS FOLDER**

The Downloads folder on an Android device running Android version 4.4.2 is located in the USERDATA partition under “/Root/media/0/Downloads”. Within this folder, files that have been downloaded outside of the Google Play Store will be found. This may contain APK files for applications downloaded from websites, pictures,
documents, etc. In this case, the malicious APK file that was placed on the phone is found.

For this analysis, the Downloads folder contains the payload sample for each sample, which were purposefully placed on the device for installation. Upon installation, the original downloads can still be found.

App Folder

The App folder on an Android device is located in the USERDATA partition under “/Root/App”. This folder will contain APK files for the applications installed on the device.

Installing Godwon on the device creates an APK to this folder called “com.xinghai.contact-2.apk.” Like the original payload in the Downloads folder, this APK file is 22435 bytes in size and has the same MD5 hash value of 0CCF75E179D91CCBD86722014F014607. The sample of Social Path also has an APK file saved to this location called “com.savemebeta-1.apk” with an MD5 hash value and file size that matches the original payload in the Downloads folder (MD5: 78835947CCA21BA42110A4F206A7A486; file size 2318602 bytes).
On Android devices, every instance of a file with the same name will be stored. These different files are organized by adding a number to the end of the filename. This holds true when applications are updated, or when different versions are installed. In Figure 24 above, there are two instances of the com.xinghai.contact APK file. This is due to an update to the application which was performed when a prompt appeared asking for permission to do so. The first version of the APK file was deleted, therefore no metadata for the first version (timestamps) are uncovered using Cellebrite’s software, although the icon with the red X represents a deleted file from the Android device.

**DATA FOLDER**

Evidence of the malware installed on the device can be found in the data folder located on the USERDATA partition under “/Root/Data”. The Data folder will contain
SQLite databases holding information and configurations for each application. Installed applications will get a subdirectory folder under the Data directory to hold the databases, with each folder being named after the application. Most applications will have a subdirectory called “databases” within the folder dedicated to the application.

The directory for the malicious com.xinghai.contact application did not contain any SQLite database data at the time the Android device was acquired. Although, XML files found under the “shared_prefs” folder can be analyzed to understand preferences and configurations for the application as well. As mentioned above, Godwon is known to save an XML file named abc.xml to store count information about how many times the malware was executed, and/or how many times data was exfiltrated from the system.

```xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <int name="count" value="2"/>
</map>
```
The Data directory in the second sample for Social Path (SaveMe) did, however, contain database information, which can commonly be found in these directories. Many database files on Android devices can be parsed using Cellebrite’s software. The database file “user_info4” contains information related to user names and passwords.

As mentioned in the Static Analysis section above, this database found for Social Path may be the one that is created to store collected information that the user enters into a form or the application in general. This database then may be sent to the remote location programmed within the class files.

**APP-LIB FOLDER**

The app-lib folder located in the USERDATA directory at “/Root/app-lib” will be the location where external libraries used by an application are stored. Each application will have its own folder to store these libraries, named after the application. In this instance, the malicious applications, com.xinhai.contact-2 and com.savemebeta-1, have their own folders, although neither contain any external libraries to be used at the time the phone was acquired.
DALVIK-CACHE FOLDER

The Dalvik-cache folder on an Android device is located in the USERDATA directory at “/Root/dalvik-cache.” This folder contains the .dex files that were executed on the device. As recalled in the Application Architecture section above, dex files are Dalvik Execution files contained within an APK (compressed) file. The dex file created for the Godwon malware is named as “data@app@com.xinghai.contact-2.apk@classes.dex” and the Social Path file is named “data@app@com.savemebeta-1.apk.” The “data@app@” portion of the dex file name represents the location that the APK file is stored on the Android device, followed by the name of the APK file run.

FIGURE 32: GODWON DALVIK-CACHE

FIGURE 33: SOCIAL PATH DALVIK-CACHE
Double-clicking the filename in the directory tree will reveal a hex representation of the file contents, which can be correlated to the analysis performed using the methods mentioned in the Static Analysis section.

CONCLUSION

Understanding malware is vital in today’s world with the numerous cyber attacks occurring around the clock worldwide. Now that cybercriminals are targeting more than just desktop devices and spreading to mobile phone attacks, there is a need to understand how to analyze these attacks and respond. Having knowledge about how the malware utilizes functions and storage on the mobile phones it’s targeting, as well as defining the purpose for the malware is pertinent to performing a successful analysis of a compromised device. It is also important to note that Android anti-virus tools do a decent job detecting malware at this time using signature-based methods, although true forensic
and incident response procedures are the recommended method for identifying and understanding these types of compromises. The analysis performed on the two samples of Android malware, Godwon and Social Path, reveals several patterns between the two tests, which may be helpful for malware forensic analysis.

**Behavioral Patterns**

As mentioned, Android malware is not known for its stealth capabilities at this stage in mobile attacks. In both circumstances, the malware was installed with the user’s knowledge and permission, as would a legitimate application. Malware still needs permissions to be granted to it by the user in order to access other locations and data on the smartphone and have a successful compromise. This can be both a positive and negative requirement for malware. For malware that does not attempt to disguise itself as a legitimate application (like Godwon), a user may be hesitant to install and accept the permission requests if he or she is paying attention to what is being installed. On the other hand, this is helpful to malware which tricks the user into thinking it is providing a legitimate service, because the user already has an interest in it being installed and functional, therefore more permissions may be willingly granted to the malware.

Although, if a user is not paying attention and accidentally installs a malicious application to the device, it can easily be detected for remediation. Android malware at this point in time do not seem to hide themselves from the user, and are displayed within the menu amongst all of the other installed applications. Additionally, any services used can easily be seen in the Application Manager.
The tests performed in the amount of time given did not show any noticeable
symptoms or other behavioral patterns of an infected device. This may be due to not
allowing enough time for the malware to run (if the malware is set to sleep for a certain
amount of time before taking noticeable actions), errors in programming that breaks some
of the functionality in the malware, or if command and control mechanisms have since
been taken offline, also taking away functionality.

**FORENSIC PATTERNS**

There are a handful of folder locations where Android malware will store files and
leave traces on a mobile phone. The first of these locations is the Downloads folder. With
many instances of malware being downloaded from external sites outside of the Google
Play Store, evidence of malicious APK files can be found in the downloads folder. Even
after installation and execution, the malware does not remove traces of its original
payload, therefore the Downloads folder is a good place to start looking for malicious
files.

Malicious APK file do not appear to treat themselves any differently than
legitimate applications. Therefore, evidence of malware can be found in any of the
locations that a genuine application data would be stored. This includes the Data folder,
App folder, App-lib folder, and Dalvik-cache folder. These folder locations are known to
store data related to applications installed on the device, and since malware does not have
any stealth anti-forensic methods built into its functions, these folder locations will hold
data related to malware infections as well.
Triage Recommendations

Multiple trial-and-error tests were performed in order to find the best method to acquire and analyze infected devices. Performing File System acquisitions of the infected Androids using Cellebrite, without rooting the device, did not grant access to the folders where application data is installed; therefore most of the malware information was not captured in those images. In addition, performing a physical extraction of the device without rooting it caused Cellebrite to throw permissions errors and cancel the image. The best, and most successful method to acquire compromised Android phones is to root the device and take a physical image. While a file system extraction will still provide most of the same data and acquire more quickly, a physical image will show deleted files which can show if the malware or system deleted anything in an attempt to hide its tracks.

Once the device is collected, it is recommended to look in the locations identified and extract out samples for further analysis using static analysis methodologies. Investigating the raw code of the malware will be the most thorough method to identifying what the malware did (or tried to do) whilst installed. While forensic methods will help identify the malware and determine how it got installed on a device, the static analysis will complete the investigator’s understanding of what happened and identify the purpose of the malware.

Future Research

The research presented can be expanded upon given more time for analysis and research. More malware samples of different types can be tested to see if they fit the same
behavioral and forensic patterns mentioned above. Over time, the sophistication of Android malware is predicted to increase; therefore the forensic patterns identified may alter, expand, or become null and void. Additionally, research can be simply expanded upon by allowing the malware to run on the device for a longer period of time, such as a month or more. More behavioral patterns can come from this, which in turn may create more forensic artifacts of the infection on the device. A topic such as mobile phone malware, and malware analysis in general, is one that should be constantly researched and tested to verify patterns and gain more knowledge as malware attacks progress over time.
WORKS CITED


Linden, J. (2015, January 6). *The Privacy tool that wasn’t: Social path malware pretends to protect your data, then steals it.* Retrieved April 27, 2015, from Lookout: https://blog.lookout.com/blog/2015/01/06/socialpath/


