DETECTION AND RECOVERY OF ANTI-FORENSIC (VAULT) APPLICATIONS ON ANDROID DEVICES

Michaila Duncan and Umit Karabiyik
Sam Houston State University
Department of Computer Science
Huntsville, TX, 77340
{mbd015, umit}@shsu.edu

ABSTRACT

A significant number of mobile device users currently employ anti-forensics applications, also known as vault or locker applications, on their mobile devices in order to hide files such as photos. Because of this, investigators are required to spend a large portion of their time manually looking at the applications installed on the device. Currently, there is no automated method of detecting these anti-forensics applications on an Android device. This work presents the creation and testing of a vault application detection system to be used on Android devices. The main goal of this work is twofold: *(i)* Detecting and reporting the presence of various vault applications installed on given Android devices, and *(ii)* recovering the files that are hidden by utilizing these vault applications. The testing of our system was performed on six different devices running different versions of Android and in various states of rootedness. The findings show that with a fairly comprehensive list of known vault applications, it is possible to provide a list of the vault applications installed on the Android device and possibly provide extracted hidden files to the investigator unless they are encrypted. Hence, our work greatly reduces the amount of time that the investigators are required to spend examining the applications on the device.

Keywords: mobile forensics, Android OS, anti-forensic, data hiding, vault applications

1. INTRODUCTION

Mobile applications that enable users to store their data securely are often referred to as "vault" or "locker" applications. These applications can look like vaults or lockers, or they can imitate like other types of applications, such as calculators, media managers, or even stock managers. These vault applications require the user to type in a certain passcode in order to access and view the data that is stored within them. There are currently hundreds of these applications available for download on the Google Play Store (Google, 2018).

The hidden personal data in vault applications can be anything from photos, videos, and documents to SMS/MMS messages, call logs, and contacts and possibly other information, depending on the specific application being used (see an example in Figure 1). Because these applications claim to offer a

higher level of security than the default storage options provided with the device, many Android users have turned to these applications to store their data, whether their data is harmless or illicit. Certainly many users, including juveniles, are using these applications to store and hide illicit pictures and videos of themselves or others.

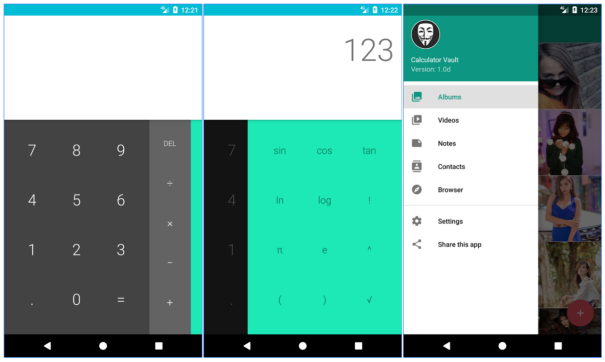

Figure 1. Screenshots of an example vault application, Calculator Vault, presenting its calculator, password entry, and hiding capabilities.

In the past two years, there has been an increase in the news of the number of police investigations that involve these vault applications, particularly involving juveniles. There have been news articles of cases in Colorado (Eckholm, 2015), Connecticut (Oliveira, 2015), Kentucky (Millward, 2017), and Huntsville, Texas (Nunez, 2017). In the criminal case in Huntsville, Texas, a juvenile student sent a recording and nude photos to another student who then sent the recording and photos to many other students. Many of the students deleted the media upon receipt, but several of them kept the media and stored it either in the default location for media on their devices or within a vault application installed on their devices. When the authorities were conducting their investigation, they found that they were required to manually examine every device in order to ensure that none of the applications installed on the devices were vault applications that contained the illicit media.

Given that the challenges have risen in the cases mentioned above, there are two main goals of this research. The first goal is to determine and report on the presence of vault applications on an Android device and provide that information to investigators. This will allow investigators to spend less time manually checking each application for hidden media, as well as increasing the probability that they will find these vault applications at all due to the fact that many of these applications look to be harmless. At the time of writing, there are at least 256 different vault applications available for download on the Google Play Store. Cellebrite's (Cellebrite, 2018) digital forensics software is used by a large number of law enforcement agencies including the United States Federal Bureau of Investigation, the United States Department of Justice, and the United States Securities and Exchange Commission (Evans, 2016). While Cellebrite's softwares currently provide many features that allow investigators to easily analyze the data acquired from mobile devices, one feature that Cellebrite's softwares are missing is reporting on the presence of vault applications on the device being acquired. Cellebrite softwares will provide a full list of the applications installed on the device, but they do not inform the investigator about suspicious applications. Because this feature does not exist, investigators currently are required to manually view the applications installed on the device to determine if these applications are on the device. The purpose of this work is to provide an automated method for investigators to utilize in order to determine if any vault applications exist on the device.

The Autopsy forensic suite is an open source forensic suite of tools that has many of the same capabilities as a commercial tool,

including mobile device forensics, but the costs of an open source tool are marginal compared to commercial tools, which makes it desirable to users who do not have the funds to purchase commercial software. Like Cellebrite, Autopsy does not currently have the capabilities to detect the presence of vault applications on an Android device.

The second goal of this study is to retrieve any unencrypted media uploaded to any vault applications found on the Android device. Not every vault application encrypts the data uploaded by the user. Because of that fact, and because a major objective of this project is to assist the investigator to recover as much as possible, as many unencrypted images as possible are retrieved and presented to the investigator in a separate Images folder.

The rest of the paper presents related work in Section 2 which is followed by the apparatus used in Section 3. Section 4 discusses the five-phase overarching methodology. The summary of findings is presented in Section 5 and Section 6 describes the conclusions and future work for the project.

2. RELATED WORK AND BACKGROUND

Related work in two major areas were reviewed. These two major areas are: Android methods of acquisition, and Android Anti-Forensic.

2.1 Android Methods of Acquisition

There are two commonly implemented methods of Android device data acquisition. These two major methods are logical acquisition and physical acquisition. In addition to these common methods, file system acquisition is another technique that is also used to acquire data from mobile devices.

Logical extraction is an extraction that stays within the confines of the file system of the device. Oftentimes, the forensic software that is performing the logical acquisition of the data will load its own mobile application onto the device. For instance, the Cellebrite UFED Forensics Software loads a device-specific API (Application Programming Interface) to the device for acquisition. After that, the API is used for read-only requests for the data in the file system of the device (Cellebrite, 2014). This method typically will not retrieve any third party or deleted data from the device; however, this method of acquisition is easier for forensic tools to organize back into the file system format (Azadegan, Yu, Liu, Sistani, & Acharya, 2012).

There are two principal types of logical acquisition: Android Debug Bridge (ADB) and Android Backup. ADB is a free command line utility included with the Android SDK that allows for a selection of actions, anything from installing applications to acquiring information from the device (Android, 2017). The ADB method of extraction can copy parts of the file system of the device. If the device is not rooted (a method of attaining root access to the phone's operating system), the ADB method of acquisition will only be selectively successful and only unencrypted app data, user data, and system information can be acquired from the device. The investigator will need to enable the hidden *Developer Options* menu in order to enable the *USB Debugging* option for the ADB acquisition method to work. If the device is rooted, the ADB method will be able to extract nearly every file and folder found in the file system of the device (Jovanovic & Redd, 2012). Android Backup acquisition occurs when a backup of the device is created and subsequently analyzed. This backup can be initiated by the forensic software (such as Cellebrite UFED Touch) at the time of ex-

amination, or the backup could have been made by the user at some point. This acquisition method retrieves little data, but even retrieving little data is better than retrieving no data at all (Cellebrite, 2015).

Physical acquisition is a method of acquisition in which a bit-by-bit copy of the device is made for analysis. This type of acquisition could potentially recover deleted data that is not possible with logical acquisition.

There are several ways to perform physical data acquisitions, but only two types will be discussed here: boot loader, and the Unix command line utility "dd." The boot loader method of physical data acquisition is performed utilizing the download mode of the device. While in this download mode, the device allows the forensic software to install a small chunk of code into the RAM of the device that is to be run during the startup, or boot-up, of the device (Cellebrite, 2014). This chunk of code is able to read and obtain a copy of the memory. This copy of the memory will then need to be interpreted by a forensic software for analysis by the investigator (see Figure 2 (Cellebrite, 2014)).

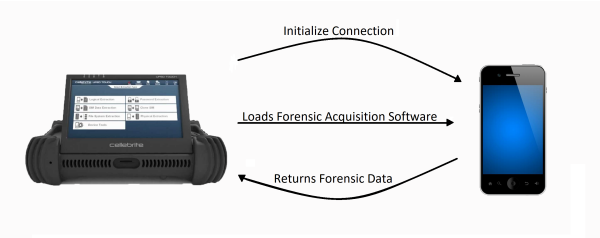

Figure 2. A boot loader is uploaded to the device that allows for the memory to be transferred to the workstation for analysis.

The "dd" method of physical acquisition utilizes the Unix and Unix-kernel operating system command line utility "dd" to copy files and folders, including copying an entire directory. For this reason, the "dd" command can be employed to copy the entire directory of an Android device to an external storage location; however, in order to utilize the "dd" command, the device must be rooted, which means that access to the root directory of the device must be granted (Lessard & Kessler, 2010). The resulting file created is referred to as a "raw image file" and can be decoded by most forensic softwares.

File system acquisition method allows making the copy of the file system of the device particularly allocated space. Although unallocated space is not acquired with this method, deleted file remnants might be identified using database files collected from the file system of the device.

2.2 Android Anti-Forensic Techniques

Anti-forensics was defined by Ryan Harris as "any attempts to compromise the availability or usefulness of evidence to the forensics process (Harris, 2006). There are four common methods of anti-forensics in digital forensics: destroying evidence, hiding evidence, eliminating evidence sources, and counterfeiting evidence (Distefano, Me, & Pace, 2010).

Destroying evidence means that the evidence has been completely eliminated and is no longer retrievable. This does not mean that the evidence was simply rendered inaccessible; it means that the evidence has been completely obliterated, whether that means that the evidence was overwritten by reformatting the device, wiping the device, or sanitizing the device. Reformatting the device is often performed by overwriting all of the ones and zeros that make up the data of the device with all zeros and then restructuring the directory of the device. Wiping the device is typically performed by overwriting the data one or more times with random data or by replacing all of the data with zeros, similar to reformatting the device, but

without restructuring the directory. This causes the data to be completely unrecoverable. Sanitizing the device can be done by overwriting the data on the device with random data (just like wiping the device, as described above), or by physically destroying the device (Garfinkel & Shelat, 2003). This method is out of the scope of this paper.

Hiding evidence is removing evidence from plain view so the investigator will be less likely to locate it. This is not always successful, but it has been found to be quite effective, especially if the evidence is placed in a location the investigator is not likely to examine, such as in an application that looks to be a simple calculator application from the outside, or by renaming the file in order to avert suspicion (Harris, 2006). This method plays on the fact that investigators are human and are not able to check every location on the device, or that the investigator may not have knowledge of the locations in which information could be hidden. This is the method of anti-forensics that is utilized by vault applications hence t0he target of our research.

Eliminating evidence sources is a method that does not destroy evidence. Instead, it prevents evidence from being created. For example, an Onion Router could be used in order to prevent Internet traffic logs from being created while communicating on a public network (Syverson, Reed, & Goldschlag, 2000).

Counterfeiting evidence is a method of anti-forensics that creates a fake collection of evidence that is meant to look like something it is not. This method of anti-forensics is meant to trick the digital forensics tools and the investigator working with the data into believing something that is not true (Harris, 2006).

Last two anti-forensics methods are also out of the scope of this paper.

3. APPARATUS

This section outlines the apparatus used in the testing of this project. All the Android devices that are utilized in this research are shown with their Android OS version and rooting options in Table 1. The initial Android device used when installing the vault applications was a Huawei Premia 4G M931 mobile phone running Android kernel version 3.0.21-perf. The service provider was MetroPCS, but there was no SIM card installed in the device. The total storage available on the device was 2.28GB total internally with 2.15GB of the internal memory available for utilization by the user. An 8GB MicroSD card was installed into the device for additional storage. The device was successfully provided root privileges using Framaroot APK and USB debugging mode was enabled. Full root privileges, which is considered to be the automatic possessing of administrator level privileges at the command prompt startup, were enabled by installing ADBD Insecure on the device.

A secondary Android device was also used for additional testing of the program. This secondary device was a LG LS840 Viper mobile phone running Android kernel version 3.0.8. The service provider was Sprint, but there was no SIM card installed in the device. The total storage available on the device was 1.73GB total internally with 0.97GB of the internal memory available for use by the user. A 16GB MicroSD card was installed into the device for additional storage. The device was successfully provided root privileges using SRSRoot and USB debugging mode was enabled. Full root privileges were enabled by installing ADBD Insecure on the device. The third device, LG Nexus 5, was used for testing as well. It was rooted using CF-Auto-Root and USB debugging mode was enabled. Full secure root privileges could not be enabled on this

Table 1. Mobile devices used in experiments.

Mobile Device	Android version	Root Option
Huawei Premia 4G M931	4.0.4 (Ice Cream Sandwich)	Rooted
LG LS840 Viper	4.0.4 (Ice Cream Sandwich)	Rooted
LG Nexus 5	6.0.1 (Marshmallow)	Rooted
Samsung Galaxy Rush SPH-M830	4.1.2 (Jelly Bean)	Unrooted
Motorola Photon Q XT897	4.0.4 (Ice Cream Sandwich)	Unrooted
Huawei Prism U8651T	2.3.6 (Gingerbread)	Unrooted

device.

A workstation running Linux Ubuntu was utilized. Tools configured on the workstation included Android Debug Bridge (ADB), Python version 2.7.12, Android Studio version 2.3.3, and an SQLite browser. A workstation running Windows 10 was also utilized. Tools configured on this workstation included Cellebrite UFED Reader version 6.2.0.79, and Autopsy forensic suite version 4.3.0. Additionally, a workstation running Windows 7 with OpenText's EnCase Software version 7.10 was utilized.

4. METHODOLOGY

In order to verify the presence of vault applications on an Android device and to retrieve the media stored within those applications, the development and testing of this research was divided into five phases: scenario creation, list creation, device analysis, program creation, and program testing.

4.1 Scenario Creation

The purpose of this phase was to simulate actual user data on the Android device. To begin with, eight different vault applications were installed on the Huawei Premia 4G M931. These applications were chosen at random from the list of 256 vault applications. A list of these applications can be seen in Table 2. The eighth application, Smart Hide Calculator (com.ids.smartcalculator) was initially installed on the device, but the application was not operational; therefore, the application was subsequently removed from the device.

Table 2. Eight initial vault applications installed on the initial rooted Android device
* Application removed from device because it was non-functional

Application Name	Package Name
Audio Manager	com.hideitpro
Calculator Private Photo Video Locker	secret.hide.calculator
Calculator Vault	com.calculator.vault
Gallery Lock	com.morrison.gallerylocklite
Keep Your Media Safe	it.ideasolutions.kyms
Photo Locker	com.handyapps.photoLocker
Vault Hide	com.netqin.ps
Smart Hide Calculator*	com.ids.smartcalculator*

After the application setup, ten different photos with the file extension ".jpg" were uploaded into each application, and no two applications received the same photos as any other application. No videos were uploaded.

4.2 List Creation

During this phase of the project, a list was created containing applications currently on the Google Play Store available for download. This list was created by performing several different searches on the Google Play Store with keywords such as "Vault", "Locker," and "Hide Media." For each application that matched the query, details of

the application, such as the full application name, the package name, and the developer name. An approximate size of the application was acquired by using a third-party website called Appbrain (AppBrain, 2018). In total, 256 different applications were entered into the list with room to add more in the future.

4.3 Device Analysis

During this phase, the Huawei Premia 4G M931 was analyzed in order to determine a course of action for the next phase of the project, which is program creation. This phase consisted of acquiring the data from the device and analyzing it to see what data was extracted from the applications installed on the device and where that data was located on the device.

The acquisition process began by utilizing a Cellebrite UFED Touch terminal. A physical acquisition, a logical acquisition, and a file system acquisition of the device was performed and each data extraction was analyzed with UFED Reader, which was installed on the Windows 10 workstation.

Table 3. Number of images acquired by each acquisition method

Acquisition Method	**Number of Images**
Physical Acquisition	1700
Logical Acquisition	600
File System Acquisition	1300

Table 3 describes the number of images acquired from the device by each data acquisition method. While examining the images located during the three acquisitions, it was determined that several of the images were located in some of the data folders associated with the anti-forensics applications installed on the device. Images being found in these folders indicates that these applications do not perform any kind of advanced hiding techniques, and instead merely remove the images from their original location.

An additional physical acquisition of the device was performed by using the Unix command line utility "dd." This device acquisition was performed with the guideline provided in (Lohrum, 2018). The acquisition was examined using Autopsy Forensic Suite installed on the Windows 10 workstation. The examination revealed that several of the images that had been initially placed onto the 8GB MicroSD Card and subsequently deleted were recovered by Autopsy. All of these images were the images uploaded to the application Audio Manager (com.hideitpro). Additional images were also located in the acquisition of the device. These images were not previously deleted, but were instead merely moved to a new location of the device storage by their respective applications. These images were the images uploaded to the applications Calculator Vault (com.calculator.vault), Calculator Private Photo Video Locker (secret.hide.calcualtor), Gallery Lock (com.morrison.gallerylocklite), and Vault Hide (com.netqin.ps). No additional images were located in the acquisitions that were useful to our work.

The Unix command line utility "dd" physical acquisition was examined using OpenText's EnCase Forensic Software (just before the time of the writing of this paper, Guidance Software, the original creator of EnCase Forensic Software, was merged with OpenText (OpenText, 2018)). The results of that examination were the same as those from the Autopsy Forensic Suite examination.

The information that was gathered from analyzing these acquisitions was useful in the creation of the program to determine the presence of anti-forensics applications. It was also helpful in order to potentially retrieve the data that was stored by the anti-forensics application.

4.4 Program Creation

This program is built with the Python programming language. It compares the package names of the applications installed on the device to the package names in the list created previously. If any of these package names are confirmed matches, the program flags that application by writing the application's name and package name to a text file, and attempts to recover any of the images hidden by each application.

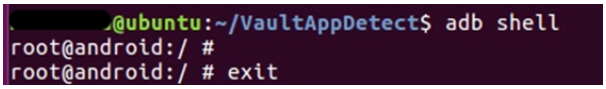

Figure 3. A fully rooted device has full root privileges at the initial command line prompt.

It was found that vault applications could be detected on three different types of Android devices: fully rooted, semi-rooted, and unrooted devices. Fully rooted devices are devices that have full root (administrator) privileges, even at the initial command line prompt (Figure 3). Semi-rooted devices are devices that are rooted, but an additional command must be executed to receive full root privileges at the command line prompt (Figure 4). Unrooted devices are devices that have no root privileges at all (Figure 5).

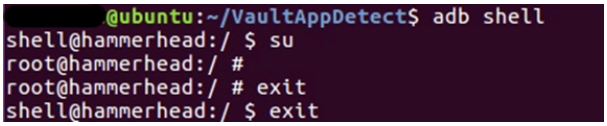

Figure 4. A semi-rooted device is rooted, but an additional command must be executed to receive full root privileges at the command line prompt.

Because fully rooted devices are the most likely devices to produce the most data, a large amount of time was spent developing a

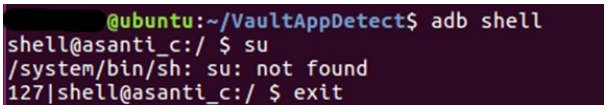

Figure 5. An unrooted device has no root privileges.

way to successfully acquire as much information as possible from the vault applications installed on this type of device. With experimentation, it was discovered that the most reliable way to detect the applications and collect data from them would be to use the following command:

```
adb pull /data/data/<Package_Name>/
```

The ADB command above will allow to access the application data files and acquire the data contained in them. This command copies the data file of an application if the package name of that application is the same as the package name of any application in the vault application list that was created previously. The copy of the data file is placed into the user-specified case directory that is placed in the original program directory. Once all of the relevant data directories are copied, the program looks for files within the directories that have the file extension of either ".jpg" or ".png" and copies those files to a separate Images directory within the user-specified case directory. As depending solely on the extension of a might be misleading, our tools also checks the header individual files for magic numbers (hexadecimal value of known file types). For example, if the first two bytes of the file header is 0xFFD8, then the file is marked as JPEG file, and if the first four bytes of the file header is 0x89504E47, then the file is marked as PNG file.

Once as many data files as possible are recovered and copied to the Images directory in the user-specified case directory, the package names associated with each of the data directories and its common application name

are copied into a text file to be reviewed by the investigator later in the event that the investigator decides that they would like to manually review the applications either on the device or with a separate forensic software, such as Cellebrite or Autopsy.

Semi-rooted devices are rooted and still have access to the data files of the device, but the command that was used for the fully rooted devices will not work for the semi-rooted devices because the administrator permissions are not immediately accessible. Therefore, an alternative method was used to detect the presence of vault applications on this type of device. To begin with, the command

```
adb shell "su -c ‘ls /data/data >
   /sdcard/Information/<FileName>.txt’"
```

was employed to compile as list of the package names of the vault applications installed on the device. Once the package names of the applications are collected, they are placed into a text file along with the common application name for the investigator to review later. The text file is saved to the user-specified case directory in the program directory. This text file is then used with the command

```
adb shell "su -c ‘cp -R /data/data/
   <PackageName>/sdcard/Information/’")
```

This command copies the package's data directory from the device's internal storage to the SD card installed to the device. That data directory can then be copied from the device to the user-specified case directory by utilizing the command

```
adb pull /sdcard/Information/<Pkg_Name>
```

Both of the commands above together are able to get around the fact that the device is only semi-rooted and requires an extra command in order to access full root capabilities.

Once the relevant data files are moved to the user-specified directory, the program follows the same procedure as that of the fully rooted devices by looking for ".jpg" and ".png" files and moving them to the Images directory of the user-specified case directory.

Many of the vault applications, such as Gallery Lock and Gallery Lock Lite, modify the data uploaded to the applications in order to circumvent attempts at recovering the data. However, many of these applications do not encrypt the data, but instead merely change the file extension. Gallery Lock and Gallery Lock Lite both perform this type of data hiding by changing the file extension to ".glk." The vault detection program is programmed to know that these two applications should have this modified data and therefore searches specifically for that data when it encounters these applications. Several applications that store their data similarly to Gallery Lock and Gallery Lock Lite are specially accounted for and the vault detection program is able to acquire their files as well. These files that have a modified file extension are typically still viewable using a standard image viewing application, so the program merely copies those files to the Image directory in the user-specified case directory in order to keep the integrity of the file intact. Additionally, many vault applications attempt to hide their data by placing the data on the device's SD Card. The vault detection program is able to search specifically for the data of several of those applications as well.

Zhang et al. (Zhang, Baggili, & Breitinger, 2017) outline their work analyzing and recovering data from the eighteen most popular vault applications available on the Google Play Store. This paper was useful in determining the different locations that a vault application can potentially use to store the uploaded images and possibly where any plaintext passwords to unlock the application could be stored on the device. Their work also provided useful information on applications that change the file extension of the uploaded data.

Table 4. Eleven additional applications and their associated package names installed to the Huawei Premia M931

Application Name	**Package Name**
Timer Lock - Photo Video Hide	krk.timerlock.timervault
Hide App, Private Dating, Safe Chat - PrivacyHider	com.trigtech.privateme
SCalc Free	com.luca1897.scalc.free
QuickPic - Photo Gallery with Google Drive Support	com.alensw.PicFolder
Magic Box (Hidden photos)	cn.menue.photohider.international
Photo Shield - Hide Photos and Videos	com.genie9.glock
Pandora	since2006.apps.pandora
Photo Gallery (Fish Bowl)	com.littlefatfish.photo
Hide Files - Andrognito	com.codexapps.andrognito
Multi Account - Secure Storage	com.reny.securespaceaccount
Undershare Photo Gallery Lock	com.minichief.undershare.viewer

Unrooted devices are more difficult to analyze because the data contained on this type of device is more protected than the data on the other two types of devices. Therefore, the program is unable to acquire the data directories of these devices. However, vault applications can still be detected on this type of device and the names of these applications can still be provided to the investigator for the investigator to review the applications manually.

In order to detect the vault applications, we utilized the following command:

```
adb shell pm list packages
```

This command lists all of the packages installed on the device. The output of this command was directed to a text file and the contents of that text file were compared against the data contained in the list of vault applications created previously. If any of the package names from the device match the package name of an application in the list, that package name and the common application name are placed into a separate text file for the investigator to review later.

For the program to know which set of commands to use on the device, it takes in user input at runtime. This input is an option that is either "r," "-s," or "u." The "-r" option selection dictates that the program should execute the fully rooted device commands. The "-s" option selection designates that the program should implement the semi-rooted device commands. The "-u" option selection means that the program should execute the unrooted device commands.

4.5 Testing and Preliminary Results

Huawei Premia 4G M931: Program testing began with the initial seven applications installed on the first rooted Android device. After the program was successful in locating, listing, and acquiring any data associated with these installed vault applications placed on the device, an additional eleven applications were installed on the first rooted Android device. Table 4 lists the names and package names of the eleven applications installed to the device. After the application installation and setup, ten different photos with the file extension ".jpg" were uploaded into each application, and no two applications received the same photos as any

other application. No videos were uploaded.

The device was fully rooted, so the program was not only able to detect the vault applications installed on the device, but was also able to recover some of the images that were uploaded to the applications. The program was able to recover all of the images from seven of the vault applications that were installed on the device. It was also able to recover some of the images from three of the other applications.

LG Viper LS840: After testing the first rooted Android device, the second fully rooted Android device was used for testing. Ten different vault applications were installed on the device. It was discovered that one of the applications, Ultimate Calculator Hide (com.chiragnew.HIDEcalculator) was not functional therefore it was subsequently removed from the device. After the initial setup for each of the applications, ten different photos with the files extension ".jpg" were uploaded into each application, and no two applications received the same photos as any other application. No videos were uploaded. Table 5 lists the names and packages names of the applications installed to the device, including the application that was removed from the device due to being non-functional.

Similar to the Huawei Premia, the LG Viper was also fully rooted. Therefore, the vault detection program was able to not only obtain the names of the vault applications installed on the device, but it was also able to recover some of the images that were uploaded to the vault applications.

The program was able to detect all nine of the installed vault applications on the device and was able to recover all of the files from one of the applications installed on the device. Our program was also able to retrieve some of the images from two additional vault applications installed on the device.

LG Nexus 5: Testing of the program's semi-rooted device vault detection capabilities was performed using the LG Nexus 5. The program was able to detect and obtain data from the installed vault applications and was able to compile the names of these applications into a text file. Table 6 lists the application names and their associated package names of the vault applications installed to the device.

The program was able to detect all ten of the installed vault applications on the device and was able to recover all of the files from three of the vault applications installed on the device. The vault detection program was also able to acquire images from two additional vault applications installed on the device.

Unrooted Devices: Each of the three unrooted devices (Samsung Galaxy Rush, Motorola Photon Q, and Huawei Prism) were tested against the program to test the capabilities of detecting vault applications on unrooted devices. Ten vault applications were installed on the Samsung Galaxy Rush (see Figure 7) and the Motorola Photon (see Figure 8) devices. Only seven applications were installed on the Huawei Prism (see Figure 9) device due to lack of storage available on the device. Ten unique photos were uploaded to each of the applications on all three of the devices.

The vault detection program was able to detect all twenty-seven applications installed on the devices and placed each application name and package name into a text file for the investigator to review later. Nevertheless, our system was not able to recover any hidden files from any of those vault applications because the vault applications keep hidden files located under */data/data* directory which is not accessible without root privileges.

Table 5. Ten applications and their associated package names installed to the LG Viper LS840
*Application removed from device because it was non-functional

Application Name	**Package Name**
Folder Lock	com.newsoftwares.folderlock_v1
Pic Lock- Hide Photos & Videos	com.xcs.piclock
A+ Gallery Photos & Videos	com.atomicadd.fotos
Private Zone - AppLock & Vault	com.leo.appmaster
Secret Gallery	com.kts.secret.gallery
Calculator Photo & Video Vault	com.yptech.privategallery
Ultimate Calculator Hide*	com.chiragnew.HIDEcalculator*
Hide Images Videos And Files	com.alpha.filehider
Vault : Hide Pictures Videos Gallery & Files	com.gamemalt.vault
Calculator Lock - Private Photo hide	com.calculatorlock

Table 6. Ten applications and their associated package names installed to the LG Nexus 5

Application Name	**Package Name**
Applock	com.domobile.applock
Hide Pictures & Videos - Vaulty	com.theronrogers.vaultyfree
Hide Pictures PhotoSafe Vault	com.slickdroid.vaultypro
Gallery lock : Photo Lock	sunstarphotomedia.photolockerpro
Hide Pictures Gallery Vault	com.mallow.hidepicturesgalleryvault
Secret AppLock	com.amazing.secreateapplock
VaultDroid - Calculator Vault	com.vaultdroid.calculatorhidephoto
Hide Photos	com.domobile.hidephotos
Gallery	com.threestar.gallery
Hide Something - Photo, Video	com.colure.app.privacygallery

5. SUMMARY OF FINDINGS

From the case studies with the different devices and a total of sixty-four applications, it is determined that 100% of the vault applications can be detected, even on unrooted devices. On fully rooted and semi-rooted devices, it is possible to not only detect 100% of the applications, but it is also possible to recover hidden files that were uploaded to the vault applications by the user. The vault detection program was able to recover unencrypted images, particularly if those images were stored in the data directory of the application on the device.

While this program was mostly only able to acquire the unencrypted files associated with the vault applications, more advanced forensic programs designed to acquire and decrypt images may be able to acquire the additional images that this program was unable to obtain. This vault detection tool's primary goal was to detect whether vault applications were present on an Android device and recover as many hidden files as possible. Our testing and preliminary results show that our program is able to recover

Table 7. Ten applications and their associated package names installed to the Samsung Glaxy Rush

Application Name	**Package Name**
Photo & Video Locker - Vault	inno.gallerylocker
Folder & File Locker, Hide Picture,Video Vault Pro	gallery.locker
Secret Lock	jacob.secret.lock
Calculator Vault	carol.calculator.calculatorvault
Joy Gallery - lock, hide, good	com.threebillion.joygallery
Secret Photo Hide -Photo Vault	com.photovault
Calculator Plus Gallery Vault Photo Video Locker	com.calculatorhide.calculatorgallerylockvault
Photo Vault Hide Pictures And Videos -KeepSecret	com.arcapps.keepsafe
Photo Locker - Hide Pictures and Videos	itianz.photolocker
File Hidder - Hide Image & Video	org.ccc.pb

Table 8. Ten applications and their associated package names installed to the Motorola Photon Q

Application Name	**Package Name**
Gallery Vault - Hide Pictures And Videos	com.thinkyeah.galleryvault
Safe Gallery (Media Lock)	ukzzang.android.gallerylocklite
Secure Gallery(Pic/Video Lock)	com.sp.smartgallery.free
Vault - Hide Pictures & Videos	net.newsoftwares.folderlockadvanced
Photo Locker	com.rand.photolocker
Photo & Video (Gallery) Lock	com.os.photo.video.gallery.lock
Calculator Vault	me.lam.calculatorvault
Hide Photos	com.domobile.hidephotos
Ami Album-Hide photos & videos	com.gionee.freya.gallery
Calculator-Gallery Locker	in.ssdevelopers.security

Table 9. Seven applications and their associated package names installed to the Huawei Prism

Application Name	**Package Name**
Hide Pictures Keep Safe Vault	com.kii.safe
AppLock	com.domobile.applock
LOCX Applock Lock Apps & Photo	com.cyou.privacysecurity
Image Locker -Hide your photos	com.inno.imagelocker
Smart Hide Calculator	com.ids.smartcalculator
Super Vault - hide pictures	com.hideimagevideo.supervault
Hide photos/videos - iLock Neo	com.capacus.neo

100% of the apps and also able to recover and reconstruct significant number of hidden files with matched hash values.

6. LIMITATIONS

Although our work shows outstanding success recovering vault applications, we are also aware of some limitations. First, the targeted mobile device screen must be unlocked and USB debugging mode must be enabled. However, because this is likely not the only tool an investigator will be employing in order to analyze the device, it is likely that he or she has physical access to the device and is able to put the device into an unlocked state and enable the USB debugging capabilities. Needless to say that, other commercial tools such as Cellebrite also require the phones being unlocked and USB debugging mode activated. Second, this study only focused on Android applications available on the Google Play Store. There are other, third-party locations from which a user can download additional applications. Third, the Google Play Store is constantly changing; additional applications are being uploaded every day. The current list of vault applications created for this work will require updating on a regular basis for the program to continue to work properly. However, it is recommended that none of the entries to the list are ever removed, even if the application no longer exists on the Google Play Store, because it is possible for a user to continue to use an application even after its removal from the Google Play Store. Fourth, this work was only tested on devices running Android operating system versions 2.3.6 to 6.0.1. Anything older or newer was not tested, and it is unknown if the program will work on any other versions. Lastly, this work is limited only to Android vault and locker applications and did not take into account other mobile operating systems, such as iOS and Windows.

7. CONCLUSION AND FUTURE WORK

As shown from the results, it is possible to detect vault applications on an Android device and recover their hidden files. It is also significantly helpful to present the findings to the investigator in an easy-to-understand manner by using a list of known vault applications. Our proof of concept design shows the possibility of these objectives. As part of the future work, we plan acquiring and decrypting the encrypted data created by vault applications for presentation to the investigator. We also would like to extend our work to all other file types and information hidden. We also believe that the future work should also examine the possibility of detecting vault applications on iOS devices. Lastly, work should explore the feasibility of creating a graphical user interface to provide additional ease-of-use for the investigator.

As part of our efforts to support open source forensic tools community and to allow future research opportunities, our source code used in this work is available upon request and also publicly shared at `http://www.shsu.edu/uxk006/resources.html`.

REFERENCES

Android. (2017). *Android debug bridge (adb), android studio.* Retrieved from `https://developer.android.com/studio/command-line/adb.html` (Accessed: 2018-01-27)

AppBrain. (2018). *Monetize, advertise and analyze android apps.* Retrieved from `http://www.Appbrain.com` (Accessed: 2018-01-27)

Azadegan, S., Yu, W., Liu, H., Sistani, M., & Acharya, S. (2012). Novel anti-forensics approaches for smart

phones. In *System science (hicss), 2012 45th hawaii international conference on* (pp. 5424–5431).

Cellebrite. (2014). *What happens when you press that button? explaining cellebrite ufed data extraction processes.* Retrieved from http://smarterforensics.com/wp-content/uploads/2014/06/Explaining-Cellebrite-UFED-Data-Extraction-Processes-final.pdf (Accessed: 2018-01-27)

Cellebrite. (2015). *Ufed touch user manual.* Retrieved from http://www.mcsira.com/WEB/8888/NSF/Web/3128/UFED_TOUCH_USER_MANUAL%202015%20ENG.pdf (Accessed: 2018-01-27)

Cellebrite. (2018). *Ufed mobile forensics applications.*

Distefano, A., Me, G., & Pace, F. (2010). Android anti-forensics through a local paradigm. *digital investigation*, *7*, S83–S94.

Eckholm, E. (2015). *Prosecutors weigh teenage sexting: Folly or felony?* Retrieved from https://www.nytimes.com/2015/11/14/us/prosecutors-in-teenage-sexting-cases-ask-foolishness-or-a-felony.html (Accessed: 2018-01-27)

Evans, J. (2016). *Cell phone forensics: Powerful tools wielded by federal investigators.* Retrieved from https://news.law.fordham.edu/jcfl/2016/06/02/cell-phone-forensics-powerful-tools-wielded-by-federal-investigators/ (Accessed: 2018-01-27)

Garfinkel, S. L., & Shelat, A. (2003). Remembrance of data passed: A study of disk sanitization practices. *IEEE Security & Privacy*, *99*(1), 17–27.

Google. (2018). *Google play store.* Retrieved from http://www.Play.google.com (Accessed: 2018-01-27)

Harris, R. (2006). Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem. *digital investigation*, *3*, 44–49.

Jovanovic, Z., & Redd, I. (2012). Android forensics techniques. *International Academy of Design and Technology*.

Lessard, J., & Kessler, G. (2010). Android forensics: Simplifying cell phone examinations.

Lohrum, M. (2018). *Free android forensics.* Retrieved from http://www.Freeandroidforensics.blogspot.com (Accessed: 2018-01-27)

Millward, E. (2017). *Six grant county teens charged after sexting scandal.* Retrieved from https://www.wcpo.com/news/local-news/grant-county/six-grant-county-middle-schoolers-arrested-after-sexting-scandal (Accessed: 2018-01-27)

Nunez, M. (2017). *Hpd investigating distribution of inappropriate digital content.* Retrieved from http://www.itemonline.com/news/local_news/hpd-investigating-distribution-of-inappropriate-digital-content/article_8f5eacb8-8534-56b6-b0a6-a4a5a6fbb49d.html (Accessed: 2018-01-27)

Oliveira, N. (2015). *Two more teens arrested in newtown sexting case.* Retrieved from http://www.newstimes.com/policereports/article/Two-more-students-arrested-in-Newtown

-sexting-case-6833636.php
(Accessed: 2018-01-27)

OpenText. (2018). *Opentext: The leader in enterprise information management.* Retrieved from https://www.opentext.com/ (Accessed: 2018-01-27)

Syverson, P. F., Reed, M. G., & Goldschlag, D. M. (2000). Onion routing access configurations. In *Darpa information survivability conference and exposition, 2000. discex'00. proceedings* (Vol. 1, pp. 34–40).

Zhang, X., Baggili, I., & Breitinger, F. (2017). Breaking into the vault: Privacy, security and forensic analysis of android vault applications. *Computers & Security*, *70*, 516–531.

