International Journal of Computer Science Trends and Technology (IJCST) – Volume 5 Issue 2, Mar – Apr 2017

RESEARCH ARTICLE OPEN ACCESS

Data Extraction Techniques for Android Based Devices

Soudamini Patil [1], Khushboo Sharma [2], Mudita Sharma [3]
Garima Chhaparwal [4], Kanchan Chowdhari [5]

Assistant Professor [1], Student [2], [3], [4], [5]
Department of Computer Science and Engineering
Cummins College of Engineering for Women
Pune – India

ABSTRACT
Hand held devices have become part and parcel in today's world. Whenever a stray device is found or seized, many a times it is challenge to identify owner of the device without possibly altering the data. This paper presents approaches to extract data in forensically sound manner which is investigation ready. The data extracted is further analyzed by various tools.

Keywords:- Data extraction, android, mobile forensics, brick, ftk, dd, imaging, adb, FTK Imager

I. INTRODUCTION

The usage of android mobile devices have become extensive in every walk of life and are misused many a times, famous examples are Taj terrorist attacks in Mumbai. Making an android device qualified to pull out data/forensic image from requires one to have the administrator access. Rooting a device gives administrator privileges but voids it from vendor warranty and is a risky task to take up with pros and cons. Further the data is extracted from SQLite databases via two methods described in the paper. First method described is android debug bridge pull and the second is imaging of a block partition using dd. Rest of the paper is organized as follows: Section II gives a brief description of rooting, adb and imaging extraction mechanism. Section III deals with rooting and how to root device. Section IV presents description of adb and how to use it to extract databases. Section V presents imaging method and viewing the image using FTK Imager toolkit. Section VI concludes the paper.

II. ROOTING, ADB, IMAGING

A. Rooting

Rooting is a process of attaining privileged control of your device (refers to android device henceforth), it grants super user permission to android sub system. The vendor specific limitations set by hardware manufacturers can be overcome by rooting.

B. adb

Android Debug Bridge is a competent command line tool that communicates with android device or the emulator. The command line provides actions such as installing or debugging apps and it provides access to UNIX shell that can be used to communicate with the device from command line and issue commands to it. The shell interface can facilitate going to device directories and seeand analyze the contents in the directory. ADB is a client server program that includes three components.

1. A client which sends command, it uses the machine you are working upon.
2. A daemon, that runs on the device you are investigating. The daemon is adbd, it runs as a background process on the device.
3. A server, which establishes communication between daemon and client. Server runs as a background process on the development machine.

To use ADB on the device connected via USB, the USB debugging option must be enabled in the developer option.

C. Imaging

To image a device/drive refers to deriving a replica of device .Image can be logical or physical. Physical image will be referred to henceforth. Imaging is a bit by bit copy of the contents including the deleted partitions. The deep insight into imaging will be discussed in section VI.

III. ROOTING AN ANDROID DEVICE

Rooting an an android device is equivalent to jail breaking, it means you have Linux like super user permissions. Gaining access to the core software of your device makes it highly vulnerable and can result into bricking the device if not done properly. By unlocking the android subsystem, the operating system, we can install the unapproved apps, replace the stock ROM by a custom ROM, update the OS, speed up the phone by overclocking and increase battery life by updating the kernel. Unlocking bootloader (which is a software at lowest level in phone) renders the device

warranty void, even if it is unrooted later , once rooted phone is very easy to recognize by manufacturer.

A. Steps to root

1. Download and install KingoRoot apk.
2. USB debugging mode has to be enabled on phone. Tap Settings, Developer Options, then tick the box for "USB debugging." Tap OK to approve the setting change.
3. Run Android Root on PC, then connect phone via its USB sync cable. The device screen may show an "Allow USB debugging?" pop-up. Tick "Always allow from this computer," and tap OK.

4. Click Root and let the utility do its work. The process can be reversed, run Android Root again, connect phone, and then click Remove Root.

IV. DATABASE EXTRACTION USING ADB

When adb is started, it checks if there is an adb server process already running. If there is not, it starts server process. When server starts it binds to local TCP port 5037 and listens for commands sent from adb client, all adb clients use adb clients use port 5037 to communicate to adb server.

A. Steps to use adb

To query via adb, the flow the commands is as follows
1. adb devices(gives the device model and ID)
2. adb root (runs the adbd daemon in root)
3. adb pull <source file path in android sub-system> <destination path in development machine>

Following table gives module names and their corresponding paths to pull from.

Table 1: Standard database path

Modules	Paths
SMS/MMS	'/data/data/com.android.providers.telephony/databaes/mmsms.db'
Default Browser	'/data/data/com.android.browser/databases/browser2.db'
Whatsapp Contacts	'/data/data/com.whatsapp/databases/wa.db'
Gmail	'/data/data/com.google.android.providers.gmail/databases/mailstore.account_name.in.com.db'
Whatsapp Calls	/data/data/com.whatsapp/databases/msgstore.db
Facebook chats	'/data/data/com.facebook.katana/databases/threads_db2'
Facebook Messenger	'/data/data/com.facebook.orca/databases/threads_db2'
WiFi Passwords	'/data/misc/wifi/wpa_supplicant.conf' from rooted extraction, or 'flattened-data' from backup extraction.
Skype Calls	'/data/data/com.skype.raider/files/<account_name>/main.db'
Synchronized Accounts	'/data/system/users/0/accounts.db'

The extracted data is stored in destination directory and opened on terminal via sqlite3 module. Below is the screenshot of one of browser database extracted along with its tables.

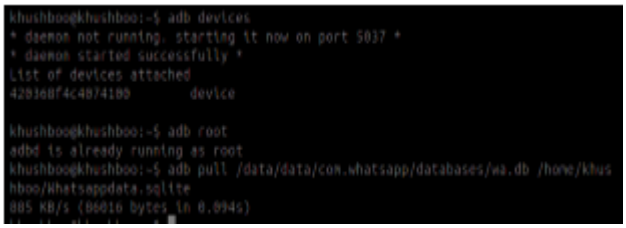


Fig 1: Pulling out database using adb

The database extracted can be viewed using SQLite prompt, android devices maintain databases in sqlite3 format. Fig. 2 shows tables and views present in browser database extracted.

Fig 2: Tables in browser database

V. IMAGING THE ANDROID DEVICE

A digital image can be a single file or collection of files , it is bit by bit representation of the block copied. It is a representation of all bits in the file from beginning to end, including the deleted space, slack space and unused space.

A. Pre - requisites to image a phone

To image a phone (which is a complete operating system as opposed to imaging a drive) , 3 things are needed

1. A connection between development machine (Linux environment here) and phone via USB.

2. An exploit to phone, android is linux based device and it has security features which disallow to take dump of the phone's storage. Using the appropriate device specific exploit, phone needs to be rooted.

3. A command to image, the command needs to be run as root which copies image one bit at a time across the USB cable, where we store it in a file.

B. Steps to image the phone

1. "adb -d shell" (this starts a shell session with the phone allowing us to interact with our phone)

2. "su" (assuming no errors, the shell starts with # i.e. root)

3. "ls /data" (check if you can access the /data directory, if yes then you have gained root permissions.

4. We use dd command (which allows to read/write block devices) to obtain bit by bit image, the command we use is device specific.

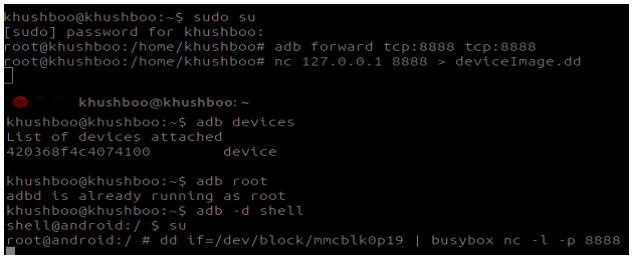
5. To find out the block to be imaged in device, open the adb shell to device and type "mount".

6. "Mount" command shows all mounted partitions on the device, check for r/w block with a familiar file system like ext4.

7. dd if=/dev/block/ xxxxxx| busybox nc -l -p 8888

8. This command writes the contents of block dev/block/xxxxxx via netcat to port 8888.

Fig 3: Terminals (adb shell and shell development machine) during imaging.



The imaging can take few minutes to complete, the process can be seen by opening destination directory in a parallel terminal. Fig 4 shows the completion of imaging process. A .dd file with appear in destination directory specified.

Fig 4: Terminal shell to device depicting completion of imaging process.

There are forensic tools to examine an image file. This investigation uses FTK imager by Access Data. To open an image in FTK go to File → Add evidence item → image. Image will be opened with all partitions visible. All packages will be visible in data partition as shown in Fig 5.

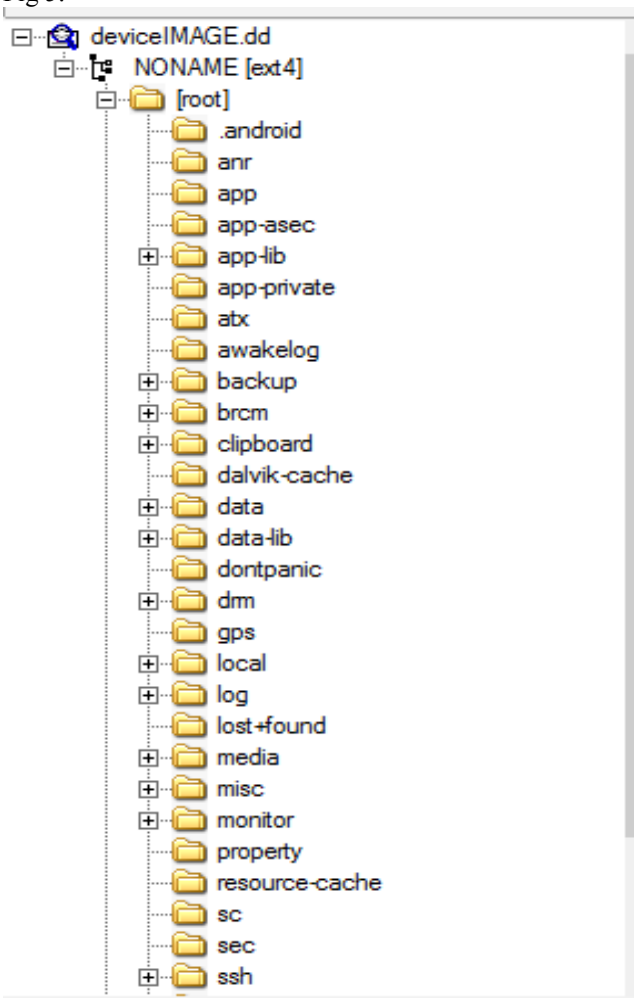


Fig 5: Directory listing of .dd image file in FTK Imager

In data directory in the listing mentioned above , userdata will be present in the form of packages.
Following screenshot shows various packages in data directory captured in FTK Imager.

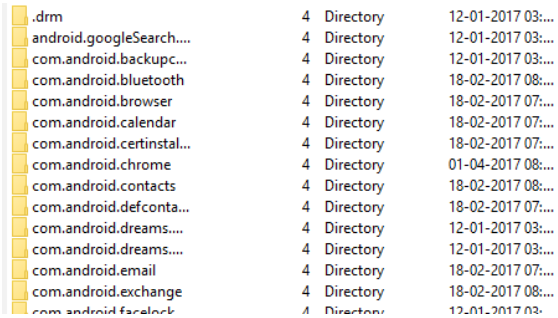


Fig 6: Packages in data directory

To export database from packages present , go to Database -> Export file. The database will be exported to destination directory. Fig 7 captures the same.

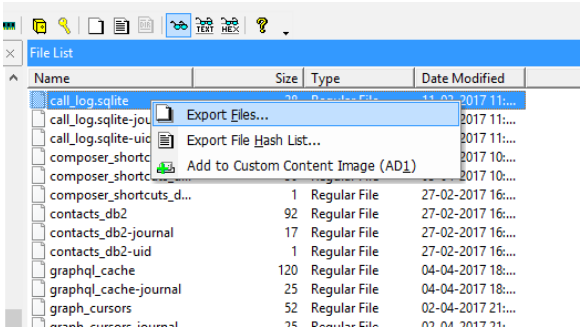


Fig 7: Exporting database from packages

VI .CONCLUSION

Thus the data extraction can be performed by various techniques, ranging from adb pull, FTP(File Transfer Protocol) and physical imaging. Every method needs a source directory to copy/transfer data from. Due to fast development in the field of mobile and technology , device specific methods need to be employed on every device to maintain it's integrity so that the data extracted is forensically sound. It is a challenge to employ one universal for all devices available.

REFERENCES

[1] Jeff Lessard and Gary C. Kessler, "Android Forensics: Simplifying Cell Phone Examinations" , Small Scale Digital Forensics Journal Vol, No. 1,September

[2] [Online] Available: https://developer.android.com/studio/command-line/adb.html

[3] [Online] Available: http://freeandroidforensics.blogspot.in/2014/08/imaging-android-device.html

[4] [Online] Available: http://freeandroidforensics.blogspot.in/2014/08/examining-image.html

[5] [Online] Available: https://forum.xda-developers.com/

[6] [Online] Available: http://www.andriller.com/decoders

[7] [YouTube] Available: https://www.youtube.com/watch?v=bfs25NijwwA