

8-6-2013

Forensic Analysis of WhatsApp on Android Smartphones

Neha S. Thakur

University Of New Orleans, nthakur@uno.edu

Follow this and additional works at: <http://scholarworks.uno.edu/td>

Recommended Citation

Thakur, Neha S., "Forensic Analysis of WhatsApp on Android Smartphones" (2013). *University of New Orleans Theses and Dissertations*. Paper 1706.

This Thesis is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UNO. It has been accepted for inclusion in University of New Orleans Theses and Dissertations by an authorized administrator of ScholarWorks@UNO. The author is solely responsible for ensuring compliance with copyright. For more information, please contact scholarworks@uno.edu.

Forensic Analysis of WhatsApp on Android Smartphones

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science
Information Assurance

by

Neha S Thakur

B.S University of Pune, 2003
M.S University of Pune, 2006

August 2013

Acknowledgements

The research and writing of this thesis has been one of the most significant academic challenges I ever had to face. Without the support and guidance of the following people, this study would not have been completed. It is to them that I owe my deepest gratitude.

I would sincerely like to thank my advisor, Dr. Golden G. Richard III for giving me an opportunity to conduct this thesis research under his excellent guidance. His exceptional knowledge, wisdom and understanding have inspired and motivated me. My professors, Dr. Shengru Tu and Dr. Adlai DePano for being on my thesis committee and mentoring me during my Master's degree in Computer Science at University of New Orleans.

I am thankful to my friend Thomas Sires, for initiating my interest in Linux, making sure I keep it up and help me solve technical issues. I appreciate the help from Aisha Ali-Gombe, Joe Sylve and Andrew Case who are on our Information Assurance group, for their contributions in the Information Assurance field and answering my questions.

Finally, I would like to thank my Parents and each member in my family and friends for their unconditional love and support. I specially want to thank my husband Mr. Naveen Singh, who has been my core source of strength through every phase of this effort and constantly encouraged me to do my best. His systematic approach and kind feedback towards my thesis has helped me improve. Words cannot express how grateful I am to you.

My heartfelt gratitude to my precious daughter Navya Singh, who was born during my Master's program and has spent many days without my complete attention. With her never give up attitude she has taught me perseverance to work around a problem until I succeed. Thank you for all your love my little girl, it kept me going.

Table of Contents

List of Figures	iv
Abstract	v
1. Introduction	1
2. Why WhatsApp Forensics	2
3. Related Work	3
3.1 WhatsApp Database - Hardware Acquisition	3
3.2 WhatsApp Database - Software Acquisition and Analysis	3
4. Android Applications	6
4.1 Memory Management	6
4.2 Data Storage	6
5. Methodology	9
6. Forensic Analysis of WhatsApp - Our Approach	10
6.1 Data Acquisition and Analysis - Non-Volatile Memory	10
6.2 Data Acquisition and Analysis - Volatile Memory(RAM)	13
6.2.1 Memfetch - Data Acquisition	15
6.2.2 WhatsApp Data Analysis and Presentation	17
7. Experimental Setup	19
8. Testing and Results	19
9. Research Findings	22
10. Conclusion	24
References	25
Vita	26

List of Figures

- Figure 1. WhatsApp Xtract Tool output
- Figure 2. Android Application Data Storage Options
- Figure 3. WhatsApp Database Structure
- Figure 4. SQLite Browser output
- Figure 5. Software Used
- Sample Result – 1
- Sample Result – 2
- Sample Result – 3
- Figure 6. Artifacts Found

Abstract

Android forensics has evolved over time offering significant opportunities and exciting challenges. On one hand, being an open source platform Android is giving developers the freedom to contribute to the rapid growth of the Android market whereas on the other hand Android users may not be aware of the security and privacy implications of installing these applications on their phones. Users may assume that a password-locked device protects their personal information, but applications may retain private information on devices, in ways that users might not anticipate. In this thesis we will be concentrating on one such application called 'WhatsApp', a popular social networking application. We will be forming an outline on how forensic investigators can extract useful information from WhatsApp and from similar applications installed on an Android platform. Our area of focus is extraction and analysis of application user data from non-volatile external storage and the volatile memory (RAM) of an Android device.

Android, WhatsApp, Volatile Memory, Mobile Forensics, Information Assurance, Application Security, Data Security, Computer Science

1. Introduction

According to Wikipedia[1] “**WhatsApp Messenger** is a proprietary, cross-platform instant messaging application for smart-phones. In addition to text messaging, users can send each other images, video, and audio media messages. The client software is available for Android, Blackberry OS, Blackberry 10, iOS, Series 40, Symbian (S60), and Windows Phone. WhatsApp Inc. was founded in 2009 by Brian Acton and Jan Koum, both veterans of *Yahoo!*, and is based in Santa Clara, California. Competing with a number of Asian-based messaging services (like LINE, KakaoTalk, and WeChat), WhatsApp was handling ten billion messages per day as of August 2012, growing from two billion in April 2012”

“According to the *Financial Times*, WhatsApp has done to SMS on mobile phones what Skype did to international calling on landlines.”

WhatsApp has gained lights' speed in terms of a social networking application. The number of downloads for WhatsApp on Android, exceeds one hundred million. In only three years it is among the top 30 free applications and among the top five free communication applications on Google Play.

WhatsApp uses your 3G or Wi-Fi (when available) to message with friends and family. It is a lightweight around 10M sized application with an option to send and receive unlimited messages for free. In all its features, WhatsApp can be used as a complete solution to easy and cost effective networking on a phone. In order to use the services of 'WhatsApp', a user must provide a phone number using which internally a user account is created with user-id such as ([phone number]@s.whatsapp.net).

WhatsApp is a cross-platform mobile messaging service which can auto sync to the address book, automatically showing all the contacts using WhatsApp. There is no restriction on the length and number of messages one can exchange and no carrier IM fees apply. One does not need to install a sim-card to use WhatsApp; the only requirements are a supported phone, internet connection and storage space on the phone to download the application.

WhatsApp uses a customized version of the open standard Extensible Messaging and Presence Protocol (XMPP) to exchange data over the internet. Messages can be in the form of plain text, multimedia message like photos, audio, video, location, address book contact cards and icons.

The plethora of personal information that can be exchanged gives us a reason to look at WhatsApp through a forensic glass.

2. Why WhatsApp forensics?

WhatsApp is a cross platform application with versions available for Android, BlackBerry, iPhone and Symbian operating systems. It is a widely used and universal application.

WhatsApp as an application is not phone dependent, like certain applications that are only supported by certain phones e.g. Motoblur intended for specific Motorola phones. WhatsApp is not carrier dependent, like certain applications supported only by certain service providers, e.g. The 'Verizon Cloud' application that is intended only for users with Verizon Wireless phone service. WhatsApp is not operating system dependent, like certain apps that can be installed only on certain platforms, e.g. Symbian or iOS specific applications. WhatsApp is not factory installed (inbuilt into the phone) and hopefully in the future may not be device dependent (as of now it can only be installed on smart-phones and not on tablets/pc's). On most smart-phones the WhatsApp application charges an initial installation amount but it is free for the android platform. This gives us one more reason to test the security of the application on the android platform in particular.

The main purpose of this research is to identify various data security issues in instant messaging applications on the Android platform which aid in forensic investigations. Information is stored in different formats at varied locations on the phone. Our aim is to summarize a general methodology to gather valuable information, so a standard investigation process can be followed for all similar applications. The goal and aim of this research was achieved successfully.

3. Related Work

The open source nature of the Android platform has tempted developers, hackers, and forensic analysts to dig into the fundamentals of the system. Before we discuss our research let us see some of the work done in this area.

3.1 WhatsApp Database - Hardware Acquisition

"Forensic Analysis of Instant Messenger Applications on Android Devices." *Mahajan, Aditya, M. S. Dahiya, and H. P. Sanghvi.* (2013) [3]

A UFED physical analyzer was used to analyze instant messenger applications (WhatsApp and Viber) on Android devices. In the case of WhatsApp, chat message artifacts, timestamps and names of files sent and received were found however the storage locations of those files were not found. In manual examination of “WhatsApp” application after the File System Extraction, database files (msgstore.db and wa.db) were found with details of chat sessions.

Mainly the database extraction and a good detailed analysis was done for existing messages. Analysis of RAM for the WhatsApp application data remnants was not considered and retrieval of deleted data was mentioned as a future project. We considered extracting deleted messages from RAM and have been able to do so. Also, database extraction was done using the UFED physical analyzer, whereas an un-encrypted version can be obtained if we root the phone.

3.2 WhatsApp Database - Software Acquisition and Analysis

Whatsapp Xtract 2.0 -Zena Forensics [4]

Another contribution to WhatsApp forensics was made by Francesco Picasso who wrote a tool to decrypt and organize SQLite database files in an organized HTML form. The tool works for both encrypted and decrypted database files.

The WhatsApp Database Encryption Project [5], has made known a vulnerability in the Android implementation of the AES cypher: the 192-bit key can be detected performing both static or active analysis on the software package. And the result is:

346a23652a46392b4d73257c67317e352e3372482177652c

A python script uses this key to decrypt and encrypted db file and presents the result in an HTML page. The paper implies that the same encryption key is used for all WhatsApp installations on Android. We used the Python tool to decrypt and interpret our encrypted database and it was done successfully. A snapshot of our output is presented in Figure.1

We can alternately read the database files through the 'SQLite browser' but the timestamps and representation of data is not straightforward. Another advantage of the tool is that the media content exchanged is displayed on the HTML page itself, one does not have to look into the media folder separately. The tool can be useful in comparing the data we analyze.

All the features of the tool sums it up into a useful one but once the messages are deleted from the database, the tool will not be able to retrieve and represent them. The tool can represent only static information that is present in the database. The databases on the external storage(usually SD card) is updated only periodically, leading to old data representation.

We wanted to find a way to retrieve deleted messages so we decided on Volatile memory acquisition and Analysis. Also, in order to acquire updated user information we must do live analysis on the device and acquire volatile memory for further analysis.

Section 4. gives details of Android Memory Management and Data Storage for Android Applications. On getting a general outlook of how applications process and store data on Android we carved a path for our research in terms of data acquisition strategies and analysis.

Figure 1. WhatsApp Xtract Tool output

Zena Forensics

WhatsApp Xtract

PK	Contact Name	Contact ID	Status	# Msg	# Unread Msg	Last Message
23	18	18-1370883818@g.us	N/A	N/A	N/A	2013-06-10 12:27:21
14	1	1-5@s.whatsapp.net	N/A	N/A	N/A	2013-06-10 12:20:39
13	1	1-3@s.whatsapp.net	N/A	N/A	N/A	2013-06-10 12:12:17


Chat session # 23: 1-3-1370883818

PK	Chat	Msg date	From	Msg content
6	18	8 2013-06-10 12:03:38	me	Hey androids, how are u
7	18	8 2013-06-10 12:15:08	me	N/A
8	18	8 2013-06-10 12:03:50	18	Hi there !
9	18	8 2013-06-10 12:04:05	18	Wanted to have a group chat with you two
10	18	8 2013-06-10 12:11:15	18	CONTACT: 1-3-1370883818 BEGIN:VCARD VERSION:3.0 N:;Dr Golden;; FN:Dr Golden

file:///home/nehad/development/Researchwork/WhatsappXtract/msgstore1.db.html

PK	Chat	Msg date	From	Msg content	Msg status	Media Type	Media Size
15	1	2013-06-10 12:20:39	me	Hi there I added you	4	0	0

Chat session # 13: 1-3-1370883818

PK	Chat	Msg date	From	Msg content	Msg status	Media Type	Media Size
2	1	2013-03-29 12:59:47	me	Hi Neha	5	0	0
3	1	2013-06-09 12:02:04	1	Hi both androids ! Here is a picture of the toshiba adapter.	0	0	0
4	1	2013-06-09 12:02:04	1	 Image	0	1	98011
5	1	2013-06-09 18:40:27	1	Thanks for the pic	5	0	0
13	1	2013-06-10 12:11:57	1	CONTACT: 1-3-1370883818 BEGIN:VCARD VERSION:3.0 N:;Dr Golden;; FN:Dr Golden TEL;type=WORK:(5 TEL;type=IPHONE: END:VCARD	0	4	0

4. Android Applications

Android applications are written in the Java programming language. The Android SDK tools compile the code into a .apk archive file. All the code in a single .apk file is considered to be one application and is the file that Android-powered devices use to install the application.

Once installed on a device, each Android application lives in its own security sandbox. The Android operating system is a multi-user Linux system in which each application is a different user. By default, every application runs in its own Linux process. Android starts the process when any of the application's components need to be executed, then shuts down the process when it's no longer needed or when the system must recover memory for other applications. [6][7][8]

4.1 Memory Management

The two primary types of memory present in Android devices are volatile (RAM) and non-volatile (NAND flash) memory. RAM is used to load and run the important parts of the OS, applications and data. RAM being volatile does not maintain its state once the phone is powered down. However, NAND flash memory is non-volatile and data is preserved even if the device is powered down.

Android has a unique mechanism to manage application memory. Android uses its own runtime environment and Dalvik virtual machine to create an efficient and secure application environment.

Android ensures application responsiveness by stopping and killing processes as necessary to free resources for higher-priority applications. In other words, all Android applications will remain running and in memory until the system needs its resources for other applications. This security feature has proven to be beneficial for our research.

4.2 Data Storage

A key part of the Android security model is that at installation, each application is assigned a unique Linux user and group ID and runs in its own process and Dalvik VM.

During installation, the operating system creates a specific directory for the application and allows only that application to access all the data stored in that directory. These mechanisms ensure data security at the low level as applications do not share memory, permissions or disk storage.

However, forensic analysts are primarily concerned with data pertaining to human interaction which can be extracted and analyzed by scrutinizing different data storage mediums through a forensic glass.

Android applications store data mainly in two locations, external storage and internal storage. On the external storage (SD card), the data can be stored and managed at any location whereas on the internal storage data storage is controlled by the Android APIs.

Persistent data is stored either to the NAND flash, the SD card or the network. A standard data storage structure for Applications is shown in Figure 2.

When an application is installed the data pertaining to the application is stored in

`/data/data/<package_name>`

e.g. `/data/data/com.whatsapp`

Inside this subdirectory there are a number of standard subdirectories found in many applications. Application developers usually control data that is stored in this directory. The most common subdirectories are lib, files, cache and databases.

Among the five methods of storing data on a device mentioned in Figure 2. Android Application Data Storage Options, analysts can uncover data from four of the five possible formats. Beyond the above formats, the Linux kernel and Android stack provide good information through logs, debugging, reverse engineering and other tools.

The remainder of this work is organized as follows: Section 5 presents the Methodology of our approach. Section 6 describes Forensic Analysis of WhatsApp - Our Approach. Experimental Setup and Testing is covered in Sections 7 and Section 8 and finally Section 9 and Section 10 shows our Research Findings and Conclusion.

Figure 2. Android Application Data Storage Options					
	1.Shared Preference	2.Internal Storage	3.External Storage	4.SQLite	5.Network
File Type stored	Key-Value pairs of primitive data stored in light-weight XML format	Files of different formats Developer based, no restriction	Files of different formats. No restriction	SQLite db format .db compact single cross-platform file	Config and network based files mainly. No restriction
Data Type	boolean, float, int, long, strings	Complicated Data Structures allowed	Complicated Data Structures allowed	SQLite supported data types	Complicated Data Structures allowed
Location	Usually in/data/data/com.android.phone/shared_prefs	Usually in /data/data subdirectory	/mnt/sdcard or emulated SD card on/mnt/emmc	Usually on internal storage/data/data/<packageName>/databases	Depends on network settings, info from log files in/data/data/files
Access Level	Owner can access	developer controlled unless Owner has root access	Owner can access M.S FAT32 fs, no security mechanism	May be encrypted unless Owner has root access	Network level
Forensic Use	Rich source of forensic data	Rich source of forensic data if root access	Rich source of forensic data	Rich source of forensic data	Forensic data from Java.net and android.net

5. Methodology

The primary goal of this research was to test the WhatsApp application from the forensic point of view on an Android phone. The basic approach to Android memory acquisition was kept in mind during our research and steps were taken to minimize any human footprints on the acquired data.

WhatsApp 2.9 [1] was first installed on more than one Android phone using the Google Play store. The application gets stored in the Internal Memory of the phone. Automatically the app syncs with the phone's contacts showing people already using WhatsApp.

When a phone with WhatsApp installed is turned on, the “com.whatsapp” process receives a signal to start the 'ExternalMediaManage' and 'MessageService' services which run in the background as long as the phone is on. Any messages exchanged are stored in the 'msgstore.db' and 'wa.db' which are SQLite databases. The databases are loaded into RAM for faster access of data. Typically all the content may not persist or may be overwritten due to swapping in RAM but this may not be true for Android.

Based on the Android process lifecycle, an application executes for as long as possible.[7] Android performs garbage collection on application-by- application basis and is based on the priority of a process. Unless a higher priority process needs more memory resources and RAM is full, the data may persist in memory for an extended period of time. This feature proves to be useful for us to extract WhatsApp contents from memory.

The WhatsApp application is an instant messenger service so users get notified about messages through the push-mechanism, as soon as any messages are received thus WhatsApp maintains a high priority in memory, mainly a visible process. This gives the user the convenience of continuously receiving messages in the background without having to download them from a typical web server like an email service.

6. Forensic Analysis of WhatsApp - Our Approach

Our approach is performing live analysis on an Android smartphone to extract user interaction information from the WhatsApp application.

We mainly concentrate on two areas of evidence collection:

1. WhatsApp Data acquisition and analysis from non-volatile memory
2. WhatsApp Data acquisition and analysis from Volatile Memory

6.1 Data Acquisition and Analysis - Non-Volatile Memory

WhatsApp stores its user data in a SQLite database (msgstore.db and wa.db). The location and structure of the database varies from platform to platform. Here we are concentrating on devices with the Android platform.

If one chooses not to root the device one can gain access to the backed up WhatsApp folder on the SD card. This folder mainly contains three sub-folders such as:

/sdcard/WhatsApp/Databases
/sdcard/WhatsApp/Media
/sdcard/WhatsApp/ProfilePictures

The database is present in the form of an encrypted file on the SD card at:

/sdcard/WhatsApp/Databases/msgstore.db.crypt

The WhatsApp Database Encryption Project [5], implies that the same AES with a 192-bit encryption key (346a23652a46392b4d73257c67317e352e3372482177652c) is being used for all WhatsApp installations on the Android platform.

WhatsApp Xtract [4] has introduced a basic Python script that takes an encrypted db file(msgstore.db.crypt) as input and gives a decrypted db file(msgstore.plain.db) as output.

The output file can be read using the sqlitebrowser software. We used an open source tool - WhatsAppXtract [4], in order to be able to read the information in a more human readable form and compare our results in the sqlitebrowser. With this tool one can open the media files and see the messages directly from the output HTML file shown in Figure 1.

The tool does not describe the complete WhatsApp database structure that one can see in the sqlitebrowser software. Please refer to Figure 3. for the msgstore.db structure. [4] In addition to the structure the SQL statements and data types can also be seen in the sqlitebrowser software.

Other than the encrypted database file one can also gain access to other sub-folders in the WhatsApp folder on the SD card like :

/sdcard/WhatsApp/Media
/sdcard/WhatsApp/ProfilePictures

The data in these folders is not encrypted. The Media folder contains all the media files such as Audios, Videos and Images exchanged during a chat session with the current user.

Backup of the files from the device to the SD card is made every day around 4:00 a.m. Absence of the most recent data from the decrypted database or the media folder only means that the application has not backed up the latest data into the database on the SD card for today yet.

On the other hand if one roots the device we see that the plain database files wa.db and msgstore.db , which can be found directly on the SD card at :

/data/data/com.whatsapp/databases/ msgstore.db and wa.db

Figure 3. WhatsApp Database Structure

wa.db	msgstore.db
Tables (3) <ul style="list-style-type: none"> android_metadata <ul style="list-style-type: none"> locale sqlite_sequence <ul style="list-style-type: none"> name seq wa_contacts <ul style="list-style-type: none"> id jid is_whatsapp_user is_iphone status number raw_contact_id display_name phone_type phone_label unseen_msg_count photo_ts 	Tables (3) <ul style="list-style-type: none"> chat_list <ul style="list-style-type: none"> _id key_remote_jid message_table_id messages <ul style="list-style-type: none"> _id key_remote_jid key_from_me key_id status needs_push data timestamp media_url media_mime_type media_wa_type media_size media_name latitude longitude thumb_image remote_resource received_timestamp send_timestamp receipt_server timestamp

On looking into the files using 'sqlitebrowser' software, we noticed the structure of the database. Please refer to Figure 4. SQLite Browser – msgstore.db -messages table.

The wa.db file contains the wa_contacts table which mainly consists of the contact list of users. The msgstore.db file mainly consists of the chat_list table and messages table. The chat_list table contains the list of users that have been contacted using WhatsApp and the number of messages exchanged with them and the messages table contains the messages in plain text including the

timestamps ,size of message, type of media file exchanged, size of media_file and status of message.

We found that on rooting the device one gains access to all files in the following directory. Useful forensic evidence can be extracted and analyzed from this folder on the external storage:

*/data/data/com.whatsapp/
e.g. /data/data/com.whatsapp/files*

This directory contains information about the current user's WhatsApp account. It contains sub-directories such as , Logs- giving device memory and network connection information and Avatars - containing profile pictures of all users with whom the current device has interacted with.

These files can be backed on a local machine in order avoid any footprints on the original data. Our research shows that on gaining access to a device through USB debugging enabled, personal information of a user can be compromised.

On gaining root access to a device almost all of the application data stored on the internal storage becomes public. One can obtain forensic evidence from the phone whether the device under investigation is rooted or not.

6.2 Data Acquisition and Analysis - Volatile Memory(RAM)

We saw that most of the Android Forensic research for the WhatsApp application was concentrated on contents of the internal flash NAND memory or external SD card. No research has been done related to volatile memory acquisition and analysis for WhatsApp.

Figure 4. SQLite Browser output – msgstore.db -messages table

_id	key	remote_j	key	key_id	statu	need	data	timestamp	media_url	media_mime	media wa	ty	media_size	me
1	1	-1	0	-1	-1	0		0			-1	-1		
2	2	137079572-5	1	1364579667-1	5	0	Hi Neha	364579987277			0	0		
3	3	137079572-5	0	137079572-5	0	2	Hi both androids ! Here is a picture of the toshiba adapter.	370797324000			0	0		
4	4	137079572-5	0	137079572-5	0	2		370797324000	https://mms8...image/jpeg		1	98011		
5	5	1370658827-1	0	1370658827-1	5	0	Thanks for the pic	370821227570			0	0		
6	6	1370658827-1	1	3858436695	6	0	Hey androids, how are u	370883818000			0	0	1	
7	7	1370658827-2	1	1370658827-2	6	0		370884508053			0	4		
8	8	1370883785-2	0	1370883785-2	0	0	Hi there !	370883830000			0	0		
9	9	1370883785-2	0	1370883785-2	0	0	Wanted to have a group chat with you two	370883845000			0	0		
10	10	1370883785-5	0	1370883785-5	0	0	BEGIN:VCARD	370884275000			4	0	Dr	
11	11	1370658827-3	1	1370658827-3	6	0		370883845000			0	4		
12	12	1370883785-5	0	1370883785-5	0	0	BEGIN:VCARD...	370884289000			4	0	Na	
13	13	1370883785-6	0	1370883785-6	0	0	BEGIN:VCARD	370884317000			4	0	Dr	
14	14	1370883785-6	0	1370883785-6	0	0	Hiiii	370884337000			0	0		
15	15	1370658827-4	4	1370658827-4	4	0	Hi there I added you	370884839096			0	0		
16	16	1370658827-5	4	1370658827-5	4	0	Hi neha	370884947905			0	0		
17	17	1370658827-6	4	1370658827-6	4	0		370885109536	https://mms8...audio/3gpp		2	15695	b6	
18	18	1370658827-7	4	1370658827-7	4	0	Nice track by Avril	370885147831			0	0		
19	19	1370658827-8	4	1370658827-8	4	0	You guys should hear it	370885157550			0	0		
20	20	1370658827-9	4	1370658827-9	4	0	Its fun to group chat	370885169194			0	0		
21	21	1370658827-10	4	1370658827-10	4	0	This stuff is fast	370885178782			0	0		

The volatile memory contains important information for any investigator as it has the most recent information accessed through the phone. A phone has become the lifeline of most humans and an IM application (like WhatsApp) on it speaks a lot about what a person has been doing lately. We need to get hold of all that information to do live analysis on the device, before it is rebooted and that information is lost. Now we will see how that information can be collected and analyzed in time.

We started with volatile memory extraction using LiME [9] which is a Loadable Kernel Module (LKM), that allows the acquisition of volatile memory from Linux and Linux-based devices,

such as those powered by Android. LiME is a complete tool for getting all the contents from the memory. When we used the tool to extract memory, we realized it gives a huge binary dump file of around the size of 512mb-1 GB containing all physical RAM contents. The binary dump obtained from LiME can be analyzed using 'The Volatility Framework'[10] ,which is an open source collection of tools to extract useful information from RAM dumps pertaining to processes.

For further research we opted to use a different approach for extraction of memory since we knew the target process (here WhatsApp) we are looking to investigate and did not need data related to all other processes in memory. Looking through a huge file with all RAM contents, could yield numerous false positives. Moreover, installation and extraction of data using Volatility could become cumbersome for only one specific process, here WhatsApp.

We decided to take a simpler approach for extraction of volatile memory as we did not need data related to all other processes, so we used 'Memfetch' [12]

6.2.1 Memfetch - Data Acquisition

Memfetch is a simple yet powerful open source utility that can dump all the memory of a running process, either immediately or when a fault condition is discovered and does not disrupt the operation of the running process. In order to run 'memfetch' on an Android device, some changes need to be made to it. [11][12]

Memfetch is lightweight and can be easily pushed to the phone's SD card using the Android Debug Bridge (adb).The following steps can aid in getting a forensic sound memory capture:

\$adb push memfetch /sdcard/memfetch

Once the file is copied to the SD card we check for the process id of WhatsApp using 'ps'. The 'ps' command *ps* displays information about a selection of the active processes. For this we obtain a shell on the phone and get super user access to it.

We can gain super user access by rooting the phone. Further details about rooting are mentioned in Section 8. Testing and Results

```
$adb shell
```

```
$su
```

```
#ps
```

On obtaining the process-id (pid) of the *com.whatsapp* process, we can extract its process memory using memfetch for android.

When we tried to execute memfetch directly on the SD card we realized that the SD card was mounted with 'noexec' option so we moved memfetch to another executable mount point. To create the mount point inside SD card we did the following:

```
#cd /sdcard
```

```
#mkdir -p tmp
```

```
#mount -t tmpfs tmpfs tmp
```

```
#cp memfetch tmp/
```

```
#cd tmp
```

```
#./memfetch pid of com.whatsapp
```

The above turns to be useful in two ways, one we have an executable mount point tmpfs and second all the result files that memfetch will output will be in one folder(here tmp).

Memory is dumped in a set of files - mfetch.lst describes all files, including their mapping addresses and source, map*.bin files are a copy of all non-anonymous maps (shared libraries, mostly) and mem*.bin files are anonymous memory blocks, most likely code, stack and heap segments.

From these output files we are interested in the heap of the application (WhatsApp) so we note the start and end address of the heap from mfetch.lst file and search a matching mem/map file. Usually it is the second memory region (mem-002.bin).

Our next target is to copy the heap dump (mem-002.bin) to the local machine to avoid any forensic footprints on the phone's SD card. So we copy it to the SD card from our local mount point and then pull it using adb on the phone.

\$adb pull /sdcard/mem-002.bin

The heap (mem-002.bin) a binary file, gives us a complete memory capture for the WhatsApp process which we need to further analysis. Memfetch is a safe method to acquire volatile memory of a process. Based on the way Android handles process memory described in section 4.1 we can assume that all our application data is present in the RAM when we acquire it. Further analysis of the heap has proved our assumption to be true.

6.2.2 WhatsApp Data Analysis and Presentation

Once we had the heap dump on our machine we analyzed the file with different tools and methods. Our aim was to find WhatsApp messages and user data.

We looked into the .binary file by creating a hex dump using 'xxd' and analyzing it manually. We also executed 'strings' to search for phone numbers, messages or user specific data. This process can be time consuming when one is unsure what data to look for but works well for targeted messages or phone numbers.

To make it simpler to display basic evidence we have made a tool namely 'whatsappRamXtract'. It is mainly used to extract information from the heap dump (mem-002.bin) and show it to the user in a readable format instead of having to search a huge file manually.

Our tool 'whatsappRamXtract' is a bash script which carves the useful evidence from a binary file. As of now we have covered three primary areas of focus in our tool but it is fully open source and customizable as per need. Options available are to output numbers (-num option), messages (-mess option) and SQL statements (-sql option).

Our tool takes the .binary file as input with the option of displaying numbers used in the current device through WhatsApp, Messages exchanged and SQL statements giving database evidence. We will see more details of its usage in section 8.

In addition to the text messages we retrieved, images were also carved out from the .binary file using Scalpel. [15] This tool is a powerful one that can carve various file formats from a huge binary file. It gives us the ease of configuring the file type we want to carve from our binary in turn displaying the completed result files in a folder. The good part is, RAM may contain deleted media which Scalpel can carve out efficiently, without us having to interpret a binary file.

Our take on Deleted Messages:

We want to make a special mention about this area because the user may think that a message that they deleted is out-of-sight so out-of-mind. In case of WhatsApp, if a message is deleted from the application it is removed from the database but not immediately from RAM (volatile memory).

Let's consider a scenario of a crime where the criminal is aware of the consequences, if his/her WhatsApp messages were to be read during an investigation. In an attempt to be proved innocent the criminal deletes all the chats from the application that can give any evidence. With the help of our work an investigator can perform live analysis on the device and retrieve any deleted message. The information is deleted from the database to maintain the integrity of the application but the way Android manages its memory, the information still lurks in the volatile memory.

7. Experimental Setup

In terms of hardware used for our research we used two Motorola Droid2 phones, a USB cable and a computer for analysis of data retrieved.

In terms of software and versions used please refer Figure 5. Software Used for more details.

Figure 5. Software Used

Software	Version
Android	2.2
WhatsApp	2.9
MemFetch (customized for Android)	0.05b
Universal Androot	1.6.2
Ubuntu	11.10
Oracle VM VirtualBox	4.2.12
Extension Pack	4.2.12
Guest additions	4.2.12
Android adt-bundle-linux	x86_64
SQLiteBrowser	2.0
WhatsApp Xtract	2.0
whatsappRamXtract.sh(our tool)	1.1
Scalpel	2.0

8. Testing and Results

In this section we apply our approach towards actual tests to show samples of output we retrieved during volatile memory acquisition and analysis.

First the phone has to be rooted. Universal Androot [14] is a good rooting kit and it doesn't disrupt most of the user processes. Also it does not require a reboot of the device so one can be

sure the memory is not wiped off. Universal Androot includes the su binary (superuser.apk), and it works well with Motorola phones.

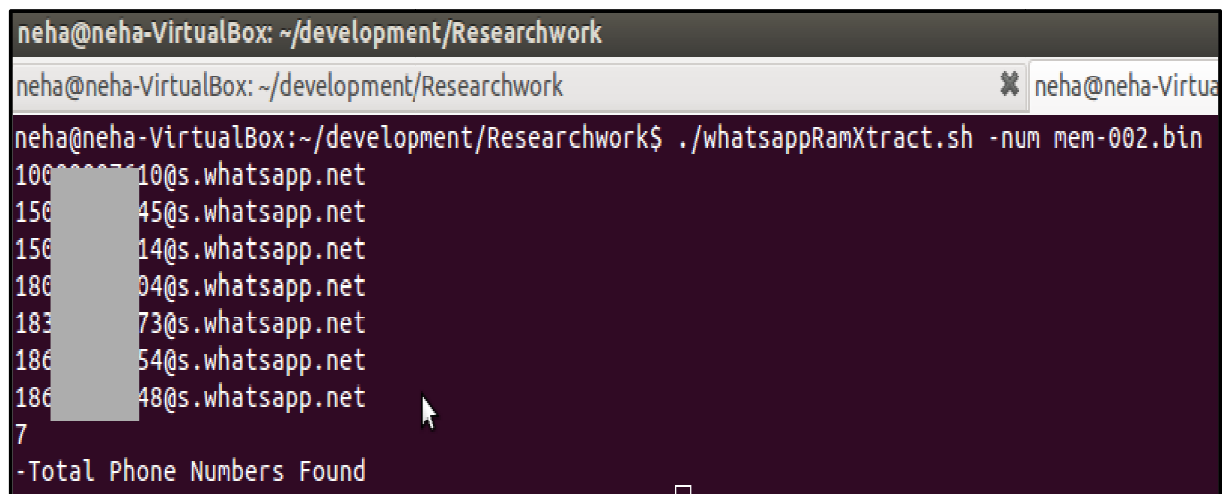
The phone should then be connected via USB, ensuring that USB debugging is turned on. A compiled memfetch is copied into the phone using the adb bridge and data is saved in the SD card. The pid for com.whatsapp, returned from ps is provided to memfetch as an argument. Execution of memfetch will return the mfetch.lst, and some *.mem and *.map files.

The file mem-002.bin is then copied using the adb bridge from the phone and whatsappRamXtract is run on it. Our tool takes three options to carve out numbers, messages or database data (SQL statements) from the file.

Following are our sample outputs on executing the tool, whatsappRamXtract against our binary file mem-002.bin.

`$/whatsappRamXtract -num mem-002.bin`

Sample Result – 1



```
neha@neha-VirtualBox: ~/development/Researchwork
neha@neha-VirtualBox: ~/development/Researchwork
neha@neha-VirtualBox:~/development/Researchwork$ ./whatsappRamXtract.sh -num mem-002.bin
1000000000@s.whatsapp.net
1500000000@s.whatsapp.net
1500000000@s.whatsapp.net
1800000000@s.whatsapp.net
1830000000@s.whatsapp.net
1860000000@s.whatsapp.net
1860000000@s.whatsapp.net
7
-Total Phone Numbers Found
```

Note : The numbers have been purposely grayed out to ensure privacy of the owner.

\$/whatsappRamXtract -mess mem-002.bin

Sample Result – 2

```
neha@neha-VirtualBox: ~/development/Researchwork
neha@neha-VirtualBox:~/development/Researchwork$ ./whatsappRamXtract.sh -mess me
m-002.bin
-----Messages exchanged-----
1936:15[REDACTED]5@s.whatsapp.net
1937-Hey there! I am using WhatsApp.50[REDACTED]45
1938-Dr Golden
--
1940:18[REDACTED]@s.whatsapp.net
1941-18[REDACTED]
1942-Verizon Wireless Accessories
--
1944:10[REDACTED]0@s.whatsapp.net
1945-00[REDACTED]
1946:18[REDACTED]4@s.whatsapp.net
1947-18[REDACTED]4 Technical Support
1948-Technical Support[
1949:18[REDACTED]4@s.whatsapp.net
1950-18[REDACTED]4
1951-#Warranty Center
--
1953:18[REDACTED]3@s.whatsapp.net
1954-Hey There !8[REDACTED]
1955-NehaNeha]
1956:18[REDACTED]@s.whatsapp.net
1957-Hey There !8[REDACTED]
1958-NehaNehav
--
1961:15[REDACTED]@s.whatsapp.net
1962-Hey there! I am using WhatsApp.5[REDACTED]
1963-Dr Golden
--
2370:1[REDACTED]3@s.whatsapp.net
2371-1[REDACTED]-52
2372-Hi both androids ! Here is a picture of the toshiba adapter.
```

\$/whatsappRamXtract -sql mem-002.bin

Sample Result – 3

```
neha@neha-VirtualBox: ~/development/Researchwork

neha@neha-VirtualBox:~/development/Researchwork$ ./whatsappRamXtract.sh -sql mem-002.bin
-----Insert Statements-----
INSERT INTO messages(key_remote_jid, key_from_me, key_id, status, needs_push, data, timestamp, media_url, media_mime_type, media_wa_type, media_size, media_name, media_hash, latitude, longitude, thumb_image, remote_resource, received_timestamp, send_timestamp, receipt_server_timestamp, receipt_device_timestamp, raw_data) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, -1, -1, -1, ?)
-----Select Statements-----
SELECT * FROM wa_contacts
SELECT * FROM wa_contacts WHERE jid IN (?, ?, ?);
SELECT * FROM wa_contacts WHERE (raw_contact_id IS NULL OR raw_contact_id=-1) AND NOT (jid LIKE '%-%')
SELECT _id FROM messages WHERE key_remote_jid=? AND key_from_me=? AND key_id=?
SELECT id, jid, is_whatsapp_user, number, raw_contact_id, display_name FROM wa_contacts WHERE is_whatsapp_user=1
-----Delete Statements-----
DELETE FILE
DELETE FROM messages WHERE _id=?
```

9. Research Findings

Our research on volatile memory has proved that critical application data is present in the RAM and can it can be extracted for further analysis. Our non-volatile external storage analysis has shown that all similar applications that load data from a SQLite database can be tested for data recovery using non-volatile memory forensics. To get easy access to data in the database, the device memory also caches it in the volatile memory. Please refer Figure 6. for more details.

We saw that data is stored in various forms by the developer and each location can become a treasure for an investigator. Rooting the device did not affect our results, in fact it gave us deeper knowledge of the directory structure of files and the data stored in them.

Figure 6. Artifacts Found			
	Non-Volatile Memory (SD Card) + No Rooting device	Non-Volatile Memory (SD Card) + Rooting device	Volatile Memory (RAM) + Rooting device
msgstore.db	Found Encrypted	Found Decrypted	Contents Found
wa.db	Not Found	Found	Contents Found
Phone Numbers	Found if db decrypted*	Found	Found
Messages	Found if db decrypted*	Found	Found
Media Files	Found if db decrypted*	Found	Found
Contact Cards	Found if db decrypted*	Found	Found
Location	Found if db decrypted*	Found	Found
SQL queries	Found if db decrypted*	Found	Found
Profile Pictures	Not Found	Found	Found
Logs	Not Found	Found	Contents Found
Directory Structure	Not Found	Found	Found
Deleted Messages	Not Found	Not Found	Found
Deleted Media Files	Not Found	Not Found	Found
Android Api's	Not Found	Not Found	Found

* indicates data that can be retrieved from 'msgstore.db' on decrypting it using the "WhatsApp Database Encryption Project Report." [5]

We were able to tap the internal storage folders, external SD card storage and RAM for constructive evidence from the WhatsApp application. Due to the process priority feature of the Android platform, application data can be retrieved from the heap. Both deleted and undeleted messages were retrieved. One thing we realized was that most of the application data gets stored on the SD card because internal storage is very small to accommodate the growing usage of social networking applications like WhatsApp. This seems to be a trend in particularly Android devices.

Here are some points to be noted that we learned during our research. It is a good idea to clear data from market app and android.services.framework services from the 'Manage Applications' settings before one downloads an application to the phone. This will ensure the cache is cleared and the market app will start successfully for application downloads.

We performed all our research inside a virtual machine which gave us an advantage to download or run executable files without having to worry about any executable affecting the host machine. Other than that all our forensic data was not leaked to the outside world and a separate environment was provided to hold all our files in one place.

USB debugging is handled differently when a virtual machine has to detect the USB attached to your system. In order to use Android Debug Bridge we had to connect our device to our machine and it needed to show up on our Virtual machine. Initially we faced the challenge that our USB connection was not getting detected.

In our case we used VirtualBox as the Virtual Machine and followed the guidelines mentioned in the VirtualBox manual [16] to 'Add USB Filter' for our Motorola phone. In addition to that we also made sure that a filter was added for every USB port of our machine by connecting the device from each port and adding it. If one happens to use a different port next time, it will not show up in the virtual machine because explicitly a filter is not added for it. After a filter was added, we logged into the Virtual Machine and made sure that '*adb devices*' shows our Motorola phone listed.

We also resized our VirtualBox image to 40 GB to be able to accommodate all our research. By default only a small partition of the disk is given to the virtual machine which did not seem sufficient for all our research.

10. Conclusion

WhatsApp has become a popular application for social networking on which people may be exchanging their personal and business related information. Our research has shown that one can gain complete access to all that information in WhatsApp and in similar social networking applications, such as ‘ Viber’. Most chat applications follow a similar pattern of storing messages in a database and updating the database periodically.

The approach we took gave a general outline for all similar applications that run on Android devices. We were able to successfully complete the aim of our research. One should be aware that a password-locked phone is not a black box and one can extract valuable application user information from the database and volatile memory.

Our research can be useful for Live Forensic Analysis on Android smart-phones. Databases are updated only once every day thus the content acquired may not be current at the time of an investigation, whereas live acquisition and analysis of the volatile memory will give us current information. When doing a forensic investigation, having the most recent messages for analysis can play a vital role. In addition to the recent messages one can look into deleted messages as well.

In the future more work can be done on interpretation of RAM data in a human readable form. Our tool can be customized to display user specific information based on one’s requirement. As of now the tool highlights three important aspects of user data namely phone numbers of users, messages they exchanged and database queries giving the basic database structure for WhatsApp.

References

- [1] [Online]. Available: <http://en.wikipedia.org/wiki/WhatsApp>
- [2] [Online]. Available: <https://play.google.com/store/apps/details?id=com.whatsapp>
- [3] Mahajan, Aditya, M. S. Dahiya, and H. P. Sanghvi. "Forensic Analysis of Instant Messenger Applications on Android Devices." arXiv preprint arXiv:1304.4915 (2013).
- [4] Zena Forensics "WhatsAppXtract 2012" [Online]. Available:
<http://code.google.com/p/hotoloti/downloads/list>
<http://blog.digital-forensics.it/2012/05/whatsapp-forensics.html>
- [5] Cortjens, D., A. Spruyt, and W. F. C. Wieringa. "WhatsApp Database Encryption Project Report."
- [6] [Online]. Available: <http://developer.android.com/guide/components/fundamentals.html>
- [7] [Online]. Available: <http://developer.android.com/guide/components/processes-and-threads.html>
- [8] Andrew Hoog : "Android Forensics, 1st Edition"
- [9] Sylve, Joseph T. "Android Memory Capture and Applications for Security and Privacy." (2011).
- [10] "Volatility," [Online]. Available: <https://www.volatilesystems.com/default/volatility>.
- [11] Ali-Gombe, Aisha Ibrahim. "Volatile Memory Message Carving: A" per process basis" Approach." (2012).
- [12] M. Zalewski, "memfetch," 2002. [Online]. Available:
<http://lcamtuf.coredump.cx/soft/memfetch.tgz>.
- [13] Vidas, T., Zhang, C., & Christin, N. (2011). Toward a general collection methodology for Android devices. digital investigation, 8, S14-S24.
- [14] Shaka, H. (2010, August 30). Universal Androot 1.6.2 beta 5. [Online]. Available:
<http://blog.23corner.com/tag/universalandroot/>
- [15] "Scalpel," [Online]. Available: <http://roussev.net/pdf/2005-DFRWS--scalpel.pdf>
- [16] [Online]. Available: <HTTP://www.virtualbox.org/manual/ch03.html#settings-usb>

Vita

Neha Singh Thakur was born in Indore, India. After she completed her work at St. Mary's school and N.Wadia College High School, Pune in 2000 she entered University of Pune in Pune, India. She received her Bachelor's degree in Computer Science in May, 2003 and her Master's degree in Computer Science in May, 2006 from University of Pune. From 2006 to 2009, she was employed as a Software Engineer with Sutra pvt. ltd. and later on as a Senior Software Engineer with Vertex Software Ltd. She got married in December 2009 in India and moved to United States with her husband. In January, 2011, she entered the Graduate School at the University of New Orleans at New Orleans LA, to pursue a Master's degree in computer science with concentration in Information Assurance. This research work was done under the guidance of Professor Golden G. Richard III in 2013.