Digital Investigation 21 (2017) 3–10

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

DFRWS USA 2016 — Proceedings of the 16th Annual USA Digital Forensics Research Conference

Forensic analysis of WeChat on Android smartphones

Songyang Wu, Yong Zhang, Xupeng Wang, Xiong Xiong*, Lin Du

The Third Research Institute of Ministry of Public Security, Shanghai 201204, China

ARTICLE INFO

Article history:
Received 13 June 2016
Received in revised form
11 November 2016
Accepted 20 November 2016
Available online 7 January 2017

Keywords:
WeChat forensics
Android smartphone forensics
Digital forensics

ABSTRACT

WeChat is one of the most popular instant-messaging smartphone applications in the world. At the end of 2015, WeChat had 697 million monthly active users from over 200 countries. Although WeChat was designed originally for communication between relatives and friends, its abundant social functions are now also used by criminals for communication, and even for the organization and coordination of criminal acts. Therefore, communication records of social networking services like WeChat extracted from the smartphones of criminals are always the vital digital evidences for the investigation and prosecution of criminal cases. At present, only a few literatures focused on WeChat forensics. This paper describes several common questions that arise in forensic examinations of Android WeChat and provides corresponding technical methods that are useful to address these questions. This paper is intended to provide vital references for the investigators and researchers working on the digital forensics.

Introduction

WeChat is one of the most popular instant-messaging (IM) smartphone applications in the world. Through the internet, users of WeChat can communicate with each other using the multimedia messages including texts, images, voices and videos. They can also use services such as WeChat Moments and Official Accounts to share and publish information. According to statistics from the annual Tencent performance reports,[1] the number of monthly active users on WeChat at the end of 2015 had reached 697 million from over 200 countries, speaking more than 20 languages. In particular, WeChat is the IM mobile application with the highest number of Chinese users.

The study of WeChat forensics has become increasingly important, as the connections have grown between social networking services (SNS) and people's daily lives. Specifically, WeChat can be used as means of communication for criminal activities, and gangs may even use its abundant social functions to organize and coordinate their criminal acts. What is more, criminals can carry out various criminal activities through WeChat, such as selling illegal items (e.g., drugs and firearms), defrauding, disseminating pornographic material to children, etc. The SNS activity records acquired from the smartphones of criminal suspects often contain evidences directly relating to illegal criminal acts. These records can play a vital role in the investigation and prosecution of cases. Therefore, it is imperative to study the investigation technical methods of SNS applications such as WeChat.

The following pictures (Fig. 1[2]) show the two basic features of WeChat. Fig. 1(a) is the chat screen and Fig. 1(b) is the Moments where users share their life with friends. WeChat messages include various types of information including text, emojis, voice, videos and images. Among these, non-text messages such as images and voices always have different meanings in different conversational contexts. Thus, forensic understanding of the conversational implications is critically dependent on whether the chat scenes (including display of all non-text messages) can be recovered completely. We use the term “scene” to represent the contexts of the chat conversation displayed as that of Fig. 1. Currently, the technical literature about WeChat forensics is relatively scarce. The mainstream digital forensic products such as Cellebrite,[3] Oxygen[4] and XRY[5] had supported Android WeChat forensics, some as early as 2013. However, the recovery of communication scene and Moments scene is limited[6] and related technical details are still not be published.

* Corresponding author.
E-mail address: xiongxiong@stars.org.cn (X. Xiong).

[1] http://www.tencent.com/zh-cn/ir/reports.shtml.

[2] http://www.wechat.com.

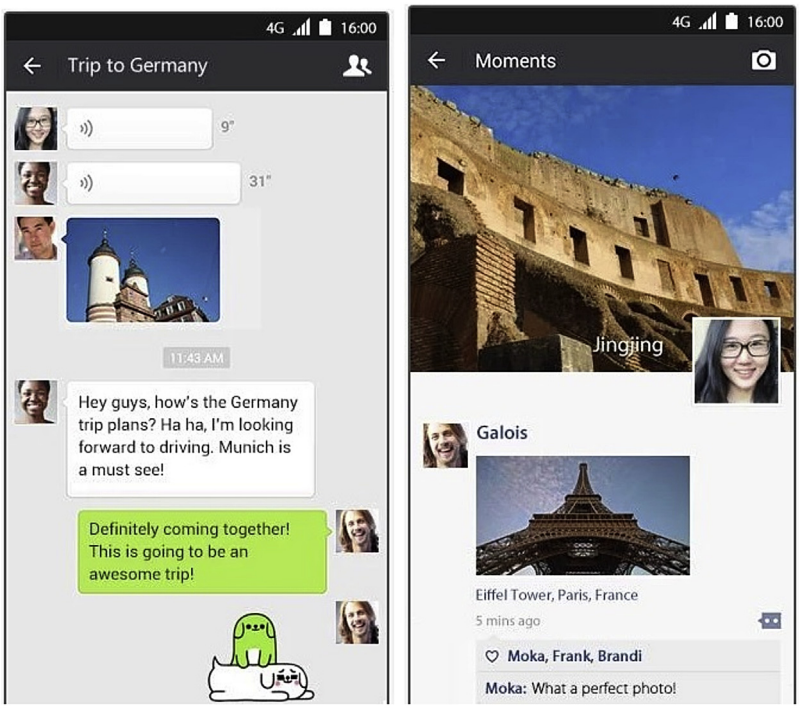
[3] http://www.cellebrite.com.

[4] http://www.oxygen-forensic.com.

[5] https://www.msab.com.

[6] We conducted tests on Cellebrite UFED v3.9, XRY v6.15, XRY v7.1 and Oxygen forensic analyst v9.0.

http://dx.doi.org/10.1016/j.diin.2016.11.002
1742-2876/



(a) The chat screen (b) Moments

Fig. 1. The two basic functionalities of WeChat.

This paper explores the common questions that arise during investigating WeChat on Android devices including: 1) how to acquire the data of WeChat and how to decode the encrypted database; 2) who did the user communicate with and what was said, and 3) what the user was sharing with the Moments. We also provide useful methods to address common challenges of WeChat forensics including conveniently backing up user data from unrooted Android devices and sufficiently recovering the scene of conversations. To evaluate the effectiveness of the proposed methods, experimental tests were conducted on different WeChat versions and Android smartphones. As far as we known, few literatures discussed forensic technology of WeChat in detail as ours. The investigators and researchers could find a better understanding in reconstructing activities related to usage of WeChat on Android smartphones.

Related works

Most studies of Android forensic technology focus on the acquisition and analysis of smartphone data obtained from flash memory and RAM. The necessary background knowledge, technology and operational method were expounded by Hoog (2011), who provided excellent guidelines for forensic workers. Vidas et al. (2011) and Simão et al. (2011) proposed a relatively universal technology for the acquisition of data from Android smartphones, involving the forensics of application data. Sylve et al. (2012) developed a tool that can dump and analyse the volatile memory of Android devices. Literatures Fairbanks (2012), Xu et al. (2013), Wu et al. (2013) developed technology to recover deleted files and SQLite records on the basis of an in-depth analysis on the file system. In general, all above schemes emphasise the acquisition and analysis of smartphone data. The forensic analysis of applications is not their focus, although they do involve some forensics of the data of third-party applications. In fact, the methods of forensic analysis for third-party applications differ due to their significant differences in implementation. Thus, the forensics of applications still requires further study.

Recently, an increasing amount of literature has considered the forensics of applications on Android smartphones. Chung et al. (2012) and Martini et al. (2015) proposed a forensic and analytical scheme for cloud applications. Mutawa et al. (2012) considered the forensic and analytical technology of three popular SNS applications, namely Facebook, Twitter and MySpace. They also performed some verification on Blackberry, iPhone and Android smartphones. Anglano (2014) put forward a method for extracting and analysing a chat history on the WhatsApp Messenger. Karpisek et al. (2015) proposed a decryption method for the network traffic of WhatsApp, as well as extraction and analytical technology for the associated communication data. On account of increasingly serious malicious software, Li et al. (2012) and Allix et al. (2014) put forward forensic technology for finding and recognizing malicious Android applications. These studies show that each application requires its own unique forensic method, and that the forensic technology in the literature cannot simply be duplicated for the forensic and analytical work on WeChat.

Zhou et al. (2015) and Silla (2015) studied the forensic technology of WeChat specifically. Zhou et al. (2015) managed to extract an encrypted and deleted chat history on WeChat by means of an in-depth analysis on the structure of the volatile Android memory. In Silla's work (Silla, 2015), the test phone was logically imaged with Android Debug Bridge (ADB) and Mobile Phone Examiner Plus (MPE+) logical tools. Validity and reliability of the test result produced by each logical tool was carefully checked by extracting logical image ten times. In their conclusion: out of the two tested tools, ADB recovered all the shared media and downloaded documents files with time stamps. Similar to Silla's work, we also logically extract WeChat data using ADB. Zhou et al. (2015) focussed mainly on the volatile memory for their WeChat forensics, while

the scheme in our paper acquires data through the file system, which is a quite different forensic approach. More importantly, this paper proposes a method for recovering the entire chat scene, and a backup acquisition method for conveniently extracting WeChat data in unrooted cases, which are issues that the above-mentioned schemes did not address well.

Several blog posts and articles[7] have been published for regarding decryption of WeChat database, they do explain the encryption scheme of Android WeChat and provide available decryption approaches. Additionally, there are two open source tools for decrypting the messages database of Android WeChat are released on the internet.[8] These works provide useful references for the development of forensic applications.

At present, commercial products for the mobile devices forensics such as Cellebrite, XRY and Oxygen all support digital investigation on various third-party Android applications and are able to extract data from unrooted Android smartphones by exploiting certain bootloader vulnerabilities that exists in many devices. These products represent the parsed chat histories in tabular form and provide investigators with related media documents and the raw SQLite databases for manually parsing. We conducted tests on Cellebrite UFED v3.9, XRY v6.15, XRY v7.1 and Oxygen forensic analyst v9.0, the recovery of communication scenes and Moments scenes is limited. No completely recovered scenes of chat or Moments may hinder a complete understanding of the conversational implications. According to recent product release notes published on the official website, Cellebrite UFED v5.0 supports decoding and parsing the communications of Android WeChat better than previous versions. However, related technical details of these products are still not published.

WeChat forensics

A necessary process of studying forensics of Android WeChat is decompiling and understanding the implementation of this APP. The available analytical tools include Apktool, dex2jar, JD-GUI, etc. The general focuses of inverse analysing of WeChat including: extracting the information about required permissions, the components and entry points of the application; identifying the key implementations including processes of encryption, the generation of encryption keys and the data-storage schemes under the help of decompiler tools such as BakSmali or JD-GUI. However, we only briefly describe reverse analysis as a background as it is not the focus of this study.

To protect the privacy of users, WeChat encrypts the database of chat messages with SQLCipher.[9] Moreover, the data acquisition through the backup functionality of Android is forbidden after the WeChat version 6.0. These results in difficulties in WeChat forensics on Android phones, that need to be firstly addressed. In this section, we start with the data acquisition of WeChat data from Android devices, the next subsection addresses decoding of the encrypted chat messages database. The remainder of this section address the most concerned problems of the investigators who conduct a digital forensic examination on an instant messaging application: 1) who did the user communicate with and what was said, and 2) what the user was sharing with the Moments.

For the purposes of study, we download version 6.2 of WeChat for Android. However the implementations including data encryption and data storage are similar in various versions of WeChat. As shown in Section Experimental results, the proposed method developed based on the version 6.2 is also available on the versions from “5.0” to “6.3.27” (the most recent WeChat version as of this writing, released on September 27, 2016).

Installation paths and data acquisition

On installation, WeChat places the application paths “/data/data/com.tencent.mm/” and “/sdcard/Tencent/MicroMsg” on Android device. The data such as chat records, configurations generated during the running of WeChat is stored in three subdirectories of the aforementioned folder “com.tencent.mm”, they are “databases”, “shared_prefs” and “MicroMsg”. The databases and shared_prefs directories cache data such as user authentication information and configuration files. The MicroMsg directory store important users' data of activities. On running, WeChat creates a number uniquely identity “uin” for each user and places a corresponding personal data folder under the path “/data/data/com.tencent.mm/MicroMsg”. The personal data folder is named using the MD5 value calculated from the user's uin. For example, the folder name could be computed as “$833dd516f909cba7ef7d16bd9a4673e8 = MD5(\text{‘mm’} + uin)$”. In the rest of this paper, the symbol $\langle udir \rangle$ is used to denote the personal folder name of a user. The path “/sdcard/Tencent/MicroMsg” is used to store the multimedia resources such as received images, audio files, etc. Each user has a private folder named also by computing $MD5(\text{‘mm’} + uin)$ under the path “/sdcard/Tencent/MicroMsg”. The most critical evidence sources under certain user folder are listed as follows:

- /data/…/⟨*udir*⟩/EnMicroMsg.db. The encrypted SQLite database of chat messages.
- /data/…/⟨*udir*⟩/SnsMicroMsg.db. SQLite database of Moments.
- /sdcard/…/⟨*udir*⟩/image2/. Raw pictures relating to the image messages.
- /sdcard/…/⟨*udir*⟩/voice2/. Raw audio files relating to the voice messages.
- /sdcard/…/⟨*udir*⟩/video/. Raw videos relating to the video messages.
- /sdcard/…/WeiXin/. Multimedia files (including images and videos) that are downloaded from Moments (through the command “Saved to phone”).

As the root privilege is required for accessing the directory “com.tencent.mm”, the acquisition of digital evidence from Android smartphones is different for rooted or unrooted devices. In the case of a rooted device, the data can be extracted directly through the Android Debug Bridge (adb) command. Specifically, this involves using the *adb pull* command to export the entire “/data/data/com.tencent.mm” directory. In the case of an unrooted Android device (i.e., one that cannot obtain root privilege), WeChat data cannot be acquired through the *adb pull* command. A new solution (called as “unrooted backup method”) was found to extract the user data of WeChat on Android device after numerous tests. For devices with WeChat versions no later than 6.0, the “backup” command provided by adb can be used to get a compressed backup file (with the .tar.gz extension) that contains the data files necessary for forensics. For devices with WeChat versions later than 6.0, the data cannot be backed up through the *adb backup* command. An alternative solution is to use the *adb install* command to degrade WeChat to version 6.0, and then use *adb backup* to back up the user data.

A naturally arisen question is that whether or not the operation downgrading the WeChat to version 6.0 causes any loss of data. We

[7] http://www.acquireforensics.com/blog/wechat-forensics-analysis.html, http://www.cnblogs.com/pieces0310/p/4216182.html and https://articles.forensicfocus.com/2014/10/01/decrypt-wechat-enmicromsgdb-database/.

[8] https://github.com/ppwwyyxx/wechat-dump and http://code.nat.moe/view/1d1d30a1.

[9] https://www.zetetic.net/sqlcipher/open-source.

carefully compared the obtained WeChat (version 6.3.27, released on 2016-09-27) data via the adb pull command with root access to that acquired through the "unrooted backup method". The test results show that 9 files such as systemInfo.cfg, staytime.cfg, etc. were modified and 3 files including wakelock_status.bin, com.android.opengl.shaders_cache and psk.key were removed. Fortunately the most important files in digital forensics such as EnMicroMsg.db, SnsMicroMsg.db, etc. were still intact.

Accessing the directory "/sdcard/Tencent/MicroMsg" does not require the root privilege, we can directly extract data of this directory via command *adb pull*.

Decrypting the messages database

EnMicroMsg.db is the SQLite database of the user's chat messages and is encrypted using the SQLCipher. We can identity, through analysing the decompiled code of WeChat APP, that the decryption key is calculated from the International Mobile Equipment Identity (IMEI) of the smartphone and the uin of current WeChat user as follows: $dec\_key = Left7(MD5(IMEI + uin))$, where the *Left*7 method extracts the first seven characters of a string. The data of IMEI and uin can be extracted from the acquired configuration files CompatibilityInfo.cfg and system_config_prefs.xml. To encrypt a SQLite database, SQLCipher splits the database file into blocks of 4 KB size and then computes the ciphertexts of these file blocks using symmetric encryption algorithm AES. Therefore, to decrypt the database file, we just need to computer plaintext of each 4 KB size block of the encrypted file using the *dec_key*.

The uin of the user is critical element for computing the decryption key. However, in case that multiple WeChat accounts logged in the same smartphone, only the uin of the last logged user is retained in the file system_config_prefs.xml. To acquire WeChat data of other users, we have to compute the uin from the name of their personal folders. The folder name of a user is $dir\_name = MD5(\text{`}mm + uin)$, where the uin is a 32 bit value. Thus we could exhaustively search the 32 bit value space and find out the uin. Exhaustive calculation always takes a long time (more than 48 h on a PC), we precomputed the all 2^{32} names of directory and stored them in a balanced binary tree. This requires about 100 GB of storage.

Communication records

As shown in Fig. 2, the chat content of WeChat often contains multimedia information, and the messages of images, emojis or voices during a chat session often convey concrete meanings like the text messages. All conversation records of the user are stored in the data table "message" of the database EnMicroMsg.db. Fig. 3 shows a fragment of table "message" that corresponds to the conversation in Fig. 2. The data fields of each message record that are interest in forensics include "talker", "create time", "type", "content", "imgPath" and "isSend". The "talker" field stores the WeChat account whom the user communicates with, his or her detailed information is stored in the data tables of "userinfo" and "rcontact". The "create time" is the time stamp of this message. Field "type" corresponds to different forms of message content, including text, images, video and audio. Different types of

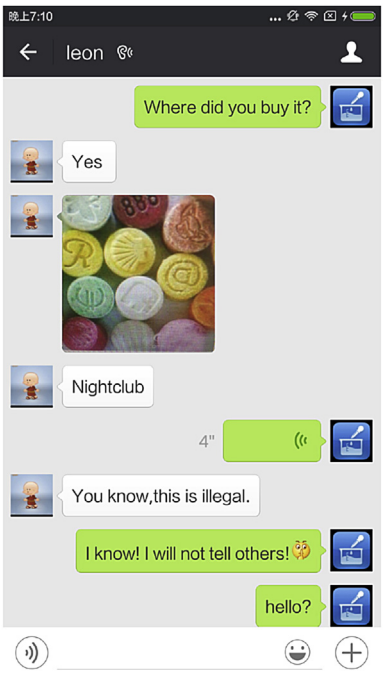


Fig. 2. A case of chat scene.

messages have different schemes of storage. The text message is stored in the field "content". Multimedia objects such as images, audio and video can be retrieved from local paths through analysing the encoded strings in the field "imgPath". If the "isSend is 1", the message is sent by the user. Otherwise, this message is created by the "talker".

It is obviously that a complete recovery of chat scene gives digital investigators a better understanding in meanings of communications. Here we explain the detail processes of retrieving the multi-media resources embed in the conversation as follows.

- For an image. The encoded string is in form as "THUMB_DIRPATH://th_dbb5e4622e87f85226c8da6893698fc0". Let *S*1 represent the header string "THUMB_DIRPATH://th_". The path of this image is computed as follows:

$$file\_path = \langle \text{uDir} \rangle + \text{“/image2/”} + substr(S1, 2, 3) + \text{“/”} + substr(S1, 6, 7) + \text{“/th\_”} + S1,$$

where uDir = "/sdcard/Tencent/MicroMsg/⟨*udir*⟩", *substr*(*S*, *start*, *end*) returns a string of the chars beginning at the start index number and running up to the end index. In this example,

$$file\_path = \langle \text{uDir} \rangle / \text{image2} / \text{b5} / \text{62} / \text{th\_dbb5e4622e87f85226c8da6893698fc0}.$$

- For an audio. We first compute the MD5 hash of the encoding string stored in the "imgPath" field. For example, let the encoding string S = 171511050616bf5274d86bf101. The path of voice message is computed as

$$file\_path = \langle \text{uDir} \rangle + \text{“/voice2/”} + substr(S1, 0, 1) + \text{“/”} + substr(S1, 3, 4) + \text{“/msg\_”} + S + \text{“.amr”},$$

RecNo	msgId	talker	content	type	creatTime	imgPath	isSend
Click here to define a filter							
39	40	wxid_i17s4u02q3ka12	oh my God,have you eaten this?	1	1462518990000	(null)	1
40	41	wxid_i17s4u02q3ka12	Where did you buy it?	1	1462519043000	(null)	1
41	42	wxid_i17s4u02q3ka12	Yes	1	1462519055000	(null)	2
42	43	wxid_i17s4u02q3ka12	<?xml version="1.0"?>	3	1462519066000	THUMBNAIL_DIRPATH://th_dbb5e4622e87f85226c8da6893698fc0	2
43	44	wxid_i17s4u02q3ka12	Nightclub	1	1462519184000	(null)	2
44	45	wxid_i17s4u02q3ka12	wxid_nxjk2ny7xjc522:3515:0	34	1462519228892	171511050616bf5274d86bf101	1
45	46	wxid_i17s4u02q3ka12	You know,this is illegal.	1	1462519294000	(null)	2
46	47	wxid_i17s4u02q3ka12	I know! I will not tell others![嘘]	1	1462519434000	(null)	1
47	48	wxid_i17s4u02q3ka12	hello?	1	1462519489000	(null)	1
48	49	wxid_i17s4u02q3ka12	You can come to my house.	1	1462519595000	(null)	2
49	51	wxid_zjvs90v9xyat12	wxid_zjvs90v9xyat12:212188:1	34	1462519604000	4515260506160dff30cada1101	2
50	52	wxid_i17s4u02q3ka12	You are now at home?	1	1462519622000	(null)	1
51	53	wxid_zjvs90v9xyat12	<?xml version="1.0"?>	3	1462519623000	THUMBNAIL_DIRPATH://th_47a728be5e0c2a1e8905eb80b42b7ce9	2
52	54	wxid_zjvs90v9xyat12	wxid_zjvs90v9xyat12:16:0	43	1462519660000	1527410605160dff30c19261	2

Fig. 3. Data table of a WeChat chat history.

where $S1 = \text{MD5}(S) =$ eb6c4a4f1efa47f290938dbd37004654. In this case, the voice is "⟨uDir⟩/voice2/ eb/6c/msg_1715110 50616bf5274d86bf101.amr".

- For a video. The storage mode of video is relatively simple. Let S be the encoding string, the path of video is

$$file\_path = \langle \text{uDir} \rangle + \text{"/video/"} + S + \text{".mp4"}.$$

After determining the specific storage paths of the multimedia files, the chat scene can be recovered successfully. Fig. 4 represents the investigation result of the communication with the talker "wxid_i17s4u02q3ka12" (described in Fig. 3) using our developed forensics tool.

Moments

The Moments is a community where users share their life with friends. Sharing records of Moments also contain multimedia messages as shown in Fig. 1(b). Messages of Moments are stored in the SnsMicroMsg.db database without encryption. The major data tables focused in forensics include "SnsInfo" and "SnsComment". The SnsInfo table stores the Moments messages, including texts and the link of multimedia files (images or videos). The SnsComment table stores the associated comments of the sharing message. Fig. 6 is an example of Moments messages stored in table "SnsInfo" that is opened directly with SQLite viewer. The corresponding fragment of Moments is shown in Fig. 5. The major focused data fields include "userName", "createTime" and "content". The "userName" field indicates the owner of the sharing message, and "content" is the data of sharing message stored as a binary large object (BLOB), as shown in Fig. 6.

The data of "content" field is consisted of multiple data segments with type-length-data (TLD) structure. In a TLD data structure, see Fig. 7(a), the first byte specifies the type of data content, the second byte indicates the data length, and the third part stores the data content itself. The detailed format of the content field of WeChat Moments can be depicted after analysing the BLOB data object. As shown in Fig. 7(b), key elements of the content field are "msgOwner", "msgContent", "msgResID", "msgImagePath" and "msgImagePath2", most of them are stored in TLD structure. The msgContent is the text of Moments message, msgOwner is the user who post the message. msgResID is an identity of the multimedia resource with 20 bytes length. The URL path of the uploaded multimedia file is stored in the msgImagePath field (Fig. 8), msgImagePath2 contains the URL of the thumbnail of the uploaded multimedia file. An efficient approach to "read" the content data is construct a customize data-analysis template (through the Text/

Fig. 4. Forensic result of our approach.

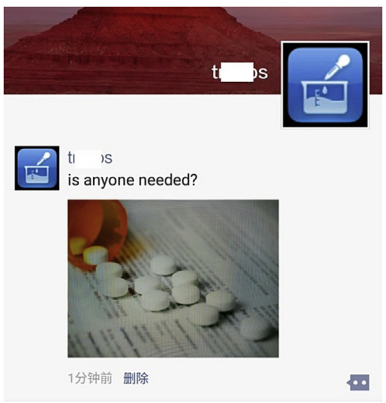


Fig. 5. A case of Moments.

RecNo	snsId	userName	localFlag	creatTime	head	localPrivate	type	sourceType	likeFlag	pravited	stringSeq	content	attrBuf	postBuf
	Click here to define a filter													
1	-6137801086392848296	wxid_nxjk2ny7xjc522	2	1467340348	20160701	0	1	2	0	0	00000123089429873167033 20			(null)
2	-6137805566153838501	wxid_i17s4u02q3ka12	2	1467339814	20160701	0	1	2	0	0	00000123089385075557131 15			(null)
3	-6137816413807431599	wxid_i17s4u02q3ka12	2	1467338521	20160701	0	1	2	0	0	00000123089276599021200 17			(null)
4	-6137858402894663461	wxid_jl3bzxqbghlj12	2	1467333515	20160701	0	3	2	0	0	00000123088856708148881 55			(null)
5	-6138356509528879017	wxid_kr088wz8knfq41	2	1467274137	20160630	0	1	2	0	0	00000123083875641806725 99			(null)
6	-6138872120280207155	wxid_jl3bzxqbghli12	2	1467212671	20160629	0	3	2	0	0	00000123078719534293444 61			(null)
7	-6139915482756935593	wxid_kr088wz8knfq41	2	1467088292	20160628	0	1	2	0	0	00000123068285909526160 23			(null)
8	-6141102535825084182	wxid_jl3bzxqbghlj12	2	1466946785	20160626	0	1	2	0	0	00000123056415378844674 34			(null)
9	-6141429987435204364	wxid_jl3bzxqbghlj12	2	1466907749	20160626	0	4	2	0	0	00000123053140862747472 52			(null)
10	-6142574697127661330	wxid_jl3bzxqbghlj12	2	1466771289	20160624	0	1	2	0	0	00000123041693765811902 86			(null)

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	
00000000	0A	14	31	32	32	36	39	33	36	32	36	33	35	39	39	31	33	36	33	38	39	37	12	13	..12269362635991363897..
00000018	77	78	69	64	5F	6E	78	6A	6B	32	6E	79	37	78	6A	63	35	32	32	18	00	20	B3	AE	wxid_nxjk2ny7xjc522.. ³®
00000030	B7	B9	05	2A	11	69	73	20	61	6E	79	6F	6E	65	20	6E	65	65	64	65	64	3F	32	1F	·¹.*.is anyone needed?2.
00000048	0D	00	00	00	00	15	00	00	00	00	1A	00	22	00	2A	00	32	00	38	00	48	00	50	00	..........".*.2.8.H.P.
00000060	58	00	65	00	00	00	00	3A	0A	0A	00	12	00	1A	00	22	00	2A	00	42	EC	02	0A	00	X.e....:.......".*.Bì...
00000078	10	01	1A	00	22	00	2A	E1	02	0A	14	31	32	32	36	39	33	36	32	36	33	36	34	31	....".*á...12269362636641
00000090	30	32	37	30	30	30	35	10	02	1A	00	22	70	68	74	74	70	3A	2F	2F	6D	6D	73	6E	0270005...."phttp://mmsn
000000A8	73	2E	71	70	69	63	2E	63	6E	2F	6D	6D	73	6E	73	2F	64	69	62	43	76	71	48	67	s.qpic.cn/mmsns/dibCvqHg
000000C0	34	57	6E	66	43	47	30	69	61	69	63	6F	73	52	69	63	73	45	39	44	73	64	42	4C	4WnfCG0iaicosRicsE9DsdBL
000000D8	62	44	32	63	64	68	68	6A	74	58	69	61	43	69	63	38	5A	6E	4C	59	30	71	5A	58	bD2cdhhjtXiaCic8ZnLY0qZX

Fig. 6. Storage of Moments.

Type (1 byte)	Length (1 byte)	Data (n bytes)

(a) Definition of TLD format

(TLD) Header	(TLD) MsgOwner	8 bytes ……	(TLD) msgContent	(TLD) msgProperty	26 bytes ……
20bytes msgResID of the first multimedia file	4 bytes ……	(TLD) msgImagePath, URL of the first multimedia file	2 bytes ……	(TLD) msgImagePath2, URL of the thumbnail of the first multimedia file	
. . . . .					
20bytes msgResID of the *Nth* multimedia file	4 bytes ……	(TLD) msgImagePath, URL of the *Nth* multimedia file	2 bytes ……	(TLD) msgImagePath2, URL of the thumbnail of the *Nth* multimedia file	
. . . . . (Other data)					

(b) Storage format of the "content" field

Fig. 7. The data stored in a Moments message.

Edit As: Hex ▾ Run Script: tldScript.1sc ▾ ▷ Run Template: tldTemplate.bt ▾ ▷

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0070h:	(00)	2A	00	42	EC	02	0A	00	10	01	1A	00	22	00	2A	E1	(.*.Bì.....".*á
0080h:	02	0A	14	31	32	32	36	39	33	36	32	36	33	36	34	31	...1226936263641
0090h:	30	32	37	30	30	30	35	10	02	1A	00	22	70	68	74	74	0270005...."phtt
00A0h:	70	3A	2F	2F	6D	6D	73	6E	73	2E	71	70	69	63	2E	63	p://mmsns.qpic.c
00B0h:	6E	2F	6D	6D	73	6E	73	2F	64	69	62	43	76	71	48	67	n/mmsns/dibCvqHg
00C0h:	34	57	6E	66	43	47	30	69	61	69	63	6F	73	52	69	63	4WnfCG0iaicosRic
00D0h:	73	45	39	44	73	64	42	4C	62	44	32	63	64	68	68	6A	sE9DsdBLbD2cdhhj
00E0h:	74	58	69	61	43	69	63	38	5A	6E	4C	59	30	71	5A	58	tXiaCic8ZnLY0qZX
00F0h:	58	54	6C	62	66	4C	78	47	73	35	71	4B	43	69	61	61	XTlbfLxGs5qKCiaa
0100h:	73	79	69	62	33	64	30	5A	76	76	73	2F	30	28	01	32	syib3d0Zvvs/0(.2
0110h:	72	68	74	74	70	3A	2F	2F	6D	6D	73	6E	73	2E	71	70	rhttp://mmsns.qp

Template Results - tldTemplate.bt

Name	Value
struct TLDType infoID	
struct TLDType infoOwner	
char cUnknow[8]	
struct TLDType msgContent	
struct TLDType file4	
char cUnknow2[26]	:
struct TLDType msgID	
char cUnknow3[4]	
struct TLDType msgImagePath	
char cType	34 '"'
unsigned byte nLen	112
char cData[112]	http://mmsns.qpic.cn/mmsns/dibCvqHg4WnfCG0iaicosRicsE9DsdBLbD2cd

Fig. 8. The "msgImagePath" value of the content.

Flag (0x01)	1byte Length of SNSID	3bytes Data	2bytes Length of SNSContent	8bytes Data
SNSID				
SNSContent				

Fig. 9. Structure of a block in the cache file sns.block.⟨num⟩.

Hex Editor "010 Editor"[10]) after determining the storage structure of this filed, as shown in Fig. 8.

The multimedia resources can be acquired from the WeChat server after extracting the URL of the multimedia file from the msgImagePath field. The thumbnails of aforementioned multimedia resources also could be extracted from the cached files in the device. Exploring the Moments always generates cache files that are stored in the path "⟨uDir⟩/sfs/sns.block.⟨num⟩", The symbol ⟨num⟩ represents a string of numbers. The cache file with the maximum num is the complete logs of exploring Moments. The sns.block.⟨num⟩ contains multiple blocks, each block has the structure illustrated in Fig. 9. The SNSID is an identity string in the format of "Wd/f/snsu_⟨msgResID⟩" or "Wd/f/snsblur_src_⟨msgResID⟩". SNSContent is the raw binary data of the thumbnail. Therefore, we can accurately locate the thumbnail of a multimedia in the Moments message using the *msgResID* (see Fig. 7(b)).

Conversion of audio file format

Audio files of WeChat with the ".aud" extension use a customized format slightly modified from the standard formats of AMR or SILK_v3. Earlier versions of WeChat used AMR to encode and store audio files, with removing only the signature information of the AMR file. Therefore, these audio files can be decoded and played through the common decoder such as FFmpeg package by adding "#!AMR" as the file header. In the newer versions of WeChat, audio encoding is based on the SILK_v3. This is the royalty-free (RF) SILK wideband audio-encoding format provided by Skype. In order to decode and play the voice message with common multimedia players during forensics, the encoding format of audio files has to be converted. A SILK_v3 audio file could be converted to a PCM audio file using the Opus voice coder[11] with the sampling rate set at 48 kHz. Since PCM audio is similar to WAV audio, the PCM audio file can be decoded and played by common audio players by adding a WAV file header.

Experiments and evaluation

Experiment setup

In our experiment, test data sets come from artificial production were used to evaluate the applicability of our method. Major forensic functions including data acquisition, decryption, and communications investigation were tested. Data acquisition depends mainly on the specific Android device and the version of WeChat, whereas decryption and communications investigation depend only on the WeChat versions.

WeChat versions 5.0 to 6.3 were installed on six mainstream Android smartphones of different brands, and artificial data was used to test the communications investigation including recovering the text message, image message, voice message and video message. The forensic experiments was conducted on a workstation with an Intel Core i7 CPU 2.4 GHz and 16 GB RAM.

Experimental results

As shown in Table 1, our data acquisition approach was tested on six devices, and the data of different WeChat v5.0 – v6.3.27 were acquired successfully. The "Xiaomi 4C" and "Samsung Galaxy S6" are unrooted, the data of WeChat v6.1 and above versions were extracted via the "unrooted backup method".

The test results of forensic functions including decrypting messages database, parsing communication records and investigating Moments are shown in Table 2, which indicate that our method discussed in this paper are practical for investigating Android WeChat from v5.0 to v6.3.27.

Although the case using WeChat on emulator is rare, we also tested the WeChat (v6.0.0.54 and v6.3.27) forensics on two emulators, Genymotion[12] and BlueStacks.[13] The Android emulator can be viewed as an Android device with root privilege, test results on Android emulators are not different compared to that on the smartphones.

A naturally arisen question is that whether or not the operation downgrading the WeChat to version 6.0 causes any loss of data. We carefully compared the obtained WeChat (version 6.3.27) data files via the adb pull command with root privilege to that acquired through the "unrooted backup method". This test was conducted on a Xiaomi 4C smartphone. We computed the MD5 value for each extracted files and look up the inconsistencies between the two file sets. The test results are shown in Table 3. There are 9 files were modified and 3 files were removed. Fortunately most of extracted files including the important EnMicroMsg.db, SnsMicroMsg.db, etc. were still intact. Another worry is that WeChat v6.0 will not be supported on a certain version of Android system in the future. In our test on Xiaomi 4C which is updated to Android 6.0 using firmware distribution of CyanogenMod[14], the "unrooted backup method" still worked well. However, in the future work, more tests on Android v6.0 and v7.0 are required and the "unrooted backup method" is also need to be improved to meet the coming changes of WeChat and Android system.

Conclusions

In this paper, we explored several common questions that arise in forensic examinations of Android WeChat including: 1) acquisition of the user data and decoding the encrypted messages database; 2) investigating the communication, and 3) what the user was sharing with the Moments. We also provide corresponding technical methods that cater to the above questions. As far as we known, few literature that analyses forensic of WeChat in detail as ours. This study can provide significant references for investigators and researchers of digital forensics.

WeChat is updated occasionally, with a typical frequency of one minor upgrade per month. An upgrade usually means possible changes in the data storage structure and the data protection measures. Such problems are also encountered in forensics of other Android applications. Therefore the reverse analysis of Android applications, and the data-protection mechanisms, should be studied continuously to satisfy the new requirements of digital investigation. Our future works also include more careful tests of "unrooted backup method" using Android v6.0 or later and the

[10] http://www.sweetscape.com/010editor.
[11] http://www.opus-codec.org.
[12] https://www.genymotion.com.
[13] http://www.bluestacks.cn.
[14] http://www.cyanogenmod.org.

Table 1
Test results of data acquisition.

Smartphone	Version						
	5.0	5.2.1	5.4.0.51	6.0.0.54	6.1.0.66	6.2.0	6.3.27
Xiaomi 4C	✓	✓	✓	✓	✓	✓	✓
Samsung Galaxy Note4	✓	✓	✓	✓	✓	✓	✓
LG G4	✓	✓	✓	✓	✓	✓	✓
HTC One	✓	✓	✓	✓	✓	✓	✓
Moto X	✓	✓	✓	✓	✓	✓	✓
Samsung Galaxy S6 Edge	✓	✓	✓	✓	✓	✓	✓

Table 2
Test results of decryption and forensics functions.

Function	Version						
	5.0	5.2.1	5.4.0.51	6.0.0.54	6.1.0.66	6.2.0	6.3.27
Decryption	✓	✓	✓	✓	✓	✓	✓
Text message	✓	✓	✓	✓	✓	✓	✓
Image message	✓	✓	✓	✓	✓	✓	✓
Voice message	✓	✓	✓	✓	✓	✓	✓
Video message	✓	✓	✓	✓	✓	✓	✓
Moments	✓	✓	✓	✓	✓	✓	✓

Table 3
Inconsistencies after degrading WeChat v6.3.27 to v6.0

	Files
Modified	getdns.ini, push_proc_startup.xml, com.tencent.mm_preferences.xml, dcfff5a88f266e972a7fd1cf80a760ec_26031732.getdns2, MM.mmap2, crash_status_file.xml, notify_key_prefxiongx2004.xml, staytime.cfg, systemInfo.cfg
Removed	wakelock_status.bin, com.android.opengl.shaders_cache, psk.key
Emerged	config.dat, md5, manifest and input.monitor

coming new versions of WeChat for identifying potential data volatility during the downgrade process. The unrooted backup method is also need to be improved to meet the new coming changes of WeChat and Android system.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 61402117) and Information Security Special Program of National Development and Reform Commission (No. FA GAI BAN GAO JI [2015]289).

References

Allix, K., Jerome, Q., Bissyand, T.F., Klein, J., State, R., Traon, Y.L., 2014. A forensic analysis of android malware – how is malware written and how it could be detected?. In: Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual, pp. 384–393. http://dx.doi.org/10.1109/COMPSAC.2014.61.

Anglano, C., 2014. Forensic analysis of whatsapp messenger on android smartphones. Digit. Investig. 11 (3), 201–213. http://dx.doi.org/10.1016/j.diin.2014.04.003 special Issue: Embedded Forensics. http://www.sciencedirect.com/science/article/pii/S1742287614000437.

Chung, H., Park, J., Lee, S., Kang, C., 2012. Digital forensic investigation of cloud storage services. Digit. Investig. 9 (2), 81–95. http://dx.doi.org/10.1016/j.diin.2012.05.015. http://www.sciencedirect.com/science/article/pii/S1742287612000400.

Fairbanks, K.D., 2012. An analysis of ext4 for digital forensics. Digit. Investig. 9 (Suppl.), S118–S130. http://dx.doi.org/10.1016/j.diin.2012.05.010 the Proceedings of the Twelfth Annual {DFRWS} Conference 12th Annual Digital Forensics Research Conference. http://www.sciencedirect.com/science/article/pii/S1742287612000357.

Hoog, A., 2011. Android Forensics: Investigation, Analysis and Mobile Security for Google Android. Syngress Publishing.

Karpisek, F., Baggili, I., Breitinger, F., 2015. Whatsapp network forensics: decrypting and understanding the Whatsapp call signaling messages. Digit. Investig. 15, 110–118. http://dx.doi.org/10.1016/j.diin.2015.09.002 special Issue: Big Data and Intelligent Data Analysis. http://www.sciencedirect.com/science/article/pii/S1742287615000985.

Li, J., Gu, D., Luo, Y., 2012. Android malware forensics: reconstruction of malicious events. In: 2012 32nd International Conference on Distributed Computing Systems Workshops, pp. 552–558. http://dx.doi.org/10.1109/ICDCSW.2012.33.

Martini, B., Do, Q., Choo, K.-K.R., 2015. Mobile Cloud Forensics: an Analysis of Seven Popular Android Apps. arXiv preprint arXiv:1506.05533.

Mutawa, N.A., Baggili, I., Marrington, A., 2012. Forensic analysis of social networking applications on mobile devices. Digit. Investig. 9 (Suppl.), S24–S33. http://dx.doi.org/10.1016/j.diin.2012.05.007 the Proceedings of the Twelfth Annual {DFRWS} Conference 12th Annual Digital Forensics Research Conference. http://www.sciencedirect.com/science/article/pii/S1742287612000321.

Silla, C., 2015. Wechat Forensic Artifacts: Android Phone Extraction and Analysis (Master's thesis). Purdue University.

Simão, A. M. d. L., Sícoli, F.C., Melo, L. P. d., Deus, F. E. G. d., Sousa Júnior, R. T. d., 2011. Acquisition and analysis of digital evidence in android smartphones. Int. J. Forensic Comput. Sci. 1, 28C43.

Sylve, J., Case, A., Marziale, L., Richard, G.G., 2012. Acquisition and analysis of volatile memory from android devices. Digit. Investig. 8 (3), 175–184. http://dx.doi.org/10.1016/j.diin.2011.10.003. http://www.sciencedirect.com/science/article/pii/S1742287611000879.

Vidas, T., Zhang, C., Christin, N., 2011. Toward a general collection methodology for android devices. Digit. Investig. 8 (Suppl.), S14–S24. http://dx.doi.org/10.1016/j.diin.2011.05.003 the Proceedings of the Eleventh Annual {DFRWS} Conference 11th Annual Digital Forensics Research Conference. http://www.sciencedirect.com/science/article/pii/S1742287611000272.

Wu, B., Xu, M., Zhang, H., Xu, J., Ren, Y., Zheng, N., 2013. A recovery approach for SQLite history recorders from YAFFS2. In: Information and Communication Technology: International Conference, ICT-EurAsia 2013, Yogyakarta, Indonesia, March 25–29, 2013. Proceedings. Springer, Berlin Heidelberg, pp. 295–299. http://dx.doi.org/10.1007/978-3-642-36818-9_30. Ch.

Xu, M., Yang, X., Wu, B., Yao, J., Zhang, H., Xu, J., Zheng, N., 2013. A metadata-based method for recovering files and file traces from {YAFFS2}. Digit. Investig. 10 (1), 62–72. http://dx.doi.org/10.1016/j.diin.2013.02.006. http://www.sciencedirect.com/science/article/pii/S1742287613000145.

Zhou, F., Yang, Y., Ding, Z., Sun, G., 2015. Dump and analysis of android volatile memory on wechat. In: 2015 IEEE International Conference on Communications (ICC), pp. 7151–7156. http://dx.doi.org/10.1109/ICC.2015.7249467.