Retrieving deleted records from Telegram

By:
Alexandros Vasilaras, Donatos Dosis, Michael Kotsis and Panagiotis Rizomiliotis

From the proceedings of
The Digital Forensic Research Conference
DFRWS APAC 2022
Sept 28-30, 2022

Forensic Science International: Digital Investigation 43 (2022) 301447

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

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

DFRWS 2022 APAC - Proceedings of the Second Annual DFRWS APAC

Retrieving deleted records from Telegram

Alexandros Vasilaras [a, *], Donatos Dosis [b, **], Michael Kotsis [b], Panagiotis Rizomiliotis [a]

[a] Department of Informatics and Telematics, Harokopio University of Athens, 9 Omirou str., Athens, 17778, Greece
[b] Department of Informatics and Computer Engineering, University of West Attica, 28 Ag. Spyridonos str., Athens, 12243, Greece

ARTICLE INFO

Article history:

Keywords:
Telegram
SQLite forensics
Android forensics
Mobile forensics

ABSTRACT

One of the most popular instant messaging applications and platforms is Telegram. In this paper, an investigation into digital forensics on Telegram is provided, deployed on contemporary android mobile phones. Due to its structure complexity, it is very challenging for forensic tools and software to properly decode its data during a forensic examination, and, especially, when it comes to deleted records retrieval. For the analysis, a realistic scenario was implemented by exchanging thousands of messages and media files among four active Telegram users, while trying to recover content from deleted text messages and exchanged picture files. Abundance of modern and up to date forensic software and tools were utilized to verify and crosscheck the results.

1. Introduction

Telegram is a cloud-based open-source social media application launched in 2013. Telegram users can exchange not only messages and media files, but also create and run chat rooms, and private channels. Telegram protects end-users' privacy, offering, among others the ability to create secret, end-to-end encrypted and self-destructing messages and chats (Telegram, 2021). Today, Telegram is available in 155 countries with more than 550 million active users per month. The country with the most installations (over 210 million) is India, followed by Russia and Indonesia (Dean, 2021).

By default, it does not enable end-to-end (E2E) encryption in normal chats and once a message arrives at the Telegram servers, it is encrypted by using its own MTProto (v2) encryption protocol, while at rest on the servers. However, Telegram could potentially read chat data since it handles encryption and decryption of the messages at its servers. E2E encryption is supported only if the secret chat feature is chosen. These two different approaches of messages handling (normal - secret) vary the structure of the data inside the Telegram android application. By extension, this affects the analysis by most forensic tools and software.

In this paper, the ability of retrieving deleted records from contemporary android devices using Telegram application is investigated. Even though Telegram supports many different types of interactions among its users (chat groups, channels, social media) this Paper focuses only on the "normal" and "secret" instant messaging between two users. The main difference between these two chat modes is the E2E encryption implementation. To the best of this paper's researchers' knowledge, this is the first systematic research focused on deleted chat records retrieval with Telegram. The analysis assumes a realistic and demanding scenario among four different active users exchanging thousands of chat messages, both text and media files.

Telegram's android application utilizes SQLite database for storing its data. Forensically the most interesting parameters of an SQLite database are the mode of journaling, the Secure Delete, the Auto Vacuum and the presence of encryption.

The examined Telegram application version (7.9.3) released on August 2021 used WAL (Write-Ahead-Log) as Journaling mode with a checkpoint set to 1.000 pages and has the Auto Vacuum and Secure Delete modes disabled.

By using the WAL as Journaling mode, the main database is left intact by committing the changes to the separate file until a checkpoint is reached. The examination of the WAL file provides information about the most recent state of the database. For instance, recently deleted data cannot be found in the main database but might be located in the database's WAL file. Furthermore, the lack of a Vacuum mode usually means that records after their deletion remain inside the database untouched forever.

In Table 1 the vital data regarding Telegram text and picture messages are shown.

* Corresponding author.
** Corresponding author.
E-mail addresses: vasilaras@hua.gr (A. Vasilaras), cscyb2010@uniwa.gr (D. Dosis), cscyb2015@uniwa.gr (M. Kotsis), prizomil@hua.gr (P. Rizomiliotis).

https://doi.org/10.1016/j.fsidi.2022.301447

Table 1
Telegram Artifacts' Directories/Files.

Content	Directory/ File Name
Telegram Database	/data/org.telegram.messenger/files/**cache4.db**
Copies of exchanged picture-media files in normal chat	/media/0/Telegram/Telegram Images
Copies of outgoing picture-media files in normal chat	/media/0/Pictures/Telegram
Picture-media files and their thumbnails	/media/0/Android/data/org.telegram.messenger/cache

1.1. Research purposes

This research was focused on whether and under which conditions is possible to retrieve deleted records from Telegram message chats, including exchanged picture-media files. The volatility or the persistence of the dataset was investigated under three different variables/conditions: (1) the elapsed time until device's forensic acquisition, (2) the device's status (power on/off, airplane mode) and (3) the interaction with the messaging application itself (e.g., creating, reading, deleting messages). In addition, several well-known forensic tools and software were evaluated, regarding their ability and accuracy of the analyzed data.

1.2. How this paper contributes to digital forensics

This research focuses on the possibility of retrieving deleted records from contemporary android devices using Telegram application. The creation of a scenario which simulates, as closely as possible, realistic situations by exchanging thousands of messages among four active Telegram users and leads to an integrated view productions and more accurate results than other related theoretical works. The main findings of the forensic analysis of the Telegram application's artifacts on modern Android devices are summarized below:

- Data volatility and how a user's actions on both Android device and the Telegram application would affect them.
- Artifacts that could be retrieved from Android OS regarding Telegram application existing apart from application's directories.
- The differences between normal and secret chat feature.
- The behavior of the SQLite database.
- The importance of validating examinations results.

1.3. How this paper is organized

The rest of the paper is structured as follows. In Section 2, related work is referenced. In Section 3, the workflow of the experiments, the followed methodology, as well as the forensic and non-forensic tools and software used for extracting and analyzing the devices, the Telegram application and SQLite databases, are presented. Section 4 contains the Analysis of Telegram application's artifacts and the ones discovered in Android devices' OS related to deleted chat records. In Section 5, artifacts related to media files are analyzed. In Section 6, variety of other findings are reported. More precisely, in Section 6.1 the Dual Application Mode is analyzed, in Section 6.2 the findings regarding the application's cloud servers and stored data are given. In Section 6.3, software and tools capabilities in application's data Decoding/Parsing are presented. Finally, in Section 7, the paper is concluded.

2. Related work

There are many guidelines about various forensic tools and techniques regarding Mobile Forensics (Nemetz et al., 2018; Easttom, 2021a, 2021b; Gogolin, 2021; Reiber, 2019; Tamma et al., 2020). There is also previous research on SQLite structure and techniques on how to retrieve deleted records and data carving from such databases (Pawlaszczyk and Hummert, 2021) (Punja and Whiffin, 2021). Pawlaszczyk D. and Hummert C. (2021) have develop an open-source forensic tool implementing those techniques, making possible to recover deleted records from a database. There is also research about the significance to conduct thorough forensic analysis of an SQLite database in order to be aware of the value of deleted records.

Anglano et al. (2017) documented Telegram application' s file and folder structure and analyzed its database. Additional forensic analysis of Telegram's database has been done through the years (Boiko Бойко, 2018) (dfirfpi, 2020).

There are also some correlations with other IMAs like WhatsApp and Viber (Sutikno et al., 2016).

3. Methodology

Experiments Setup: For the experiments two different devices were used. An LG G6 (H870) with Android 9 (security patch dated May 2019) and a Samsung A50 (A505FN/DS) with Android 11 (security patch dated July 2021). Since the access to the paths presented in Table 1 is normally prohibited, *root* access was gained on both devices. This was achieved by unlocking the bootloader and installing a custom recovery menu (Team Win Recovery Project-TWRP) and a Magisk solution (Manager, 2021). The devices' user-data directory was wiped prior to the experiments. On both devices different sim cards were activated and the Telegram application was installed using the same version of apk (through *adb -d install* command), a new user was added, and no additional applications were installed (e.g., google accounts) to mitigate the level of "noise" and unrelated data. Four different Telegram accounts exchanged the total number of the messages. This setup is not considered a forensically sound method for actual evidence's examination but was necessary to implement a white box methodology. The topic of extracting a mobile device's full file system or physical image is out the scope of this research.

Experiments Methodology: The experiments were divided into three phases as depicted in the workflow (Fig. 1). Namely, the reconnaissance phase, the main phase and the last phase. In the first phase an identification of general information about

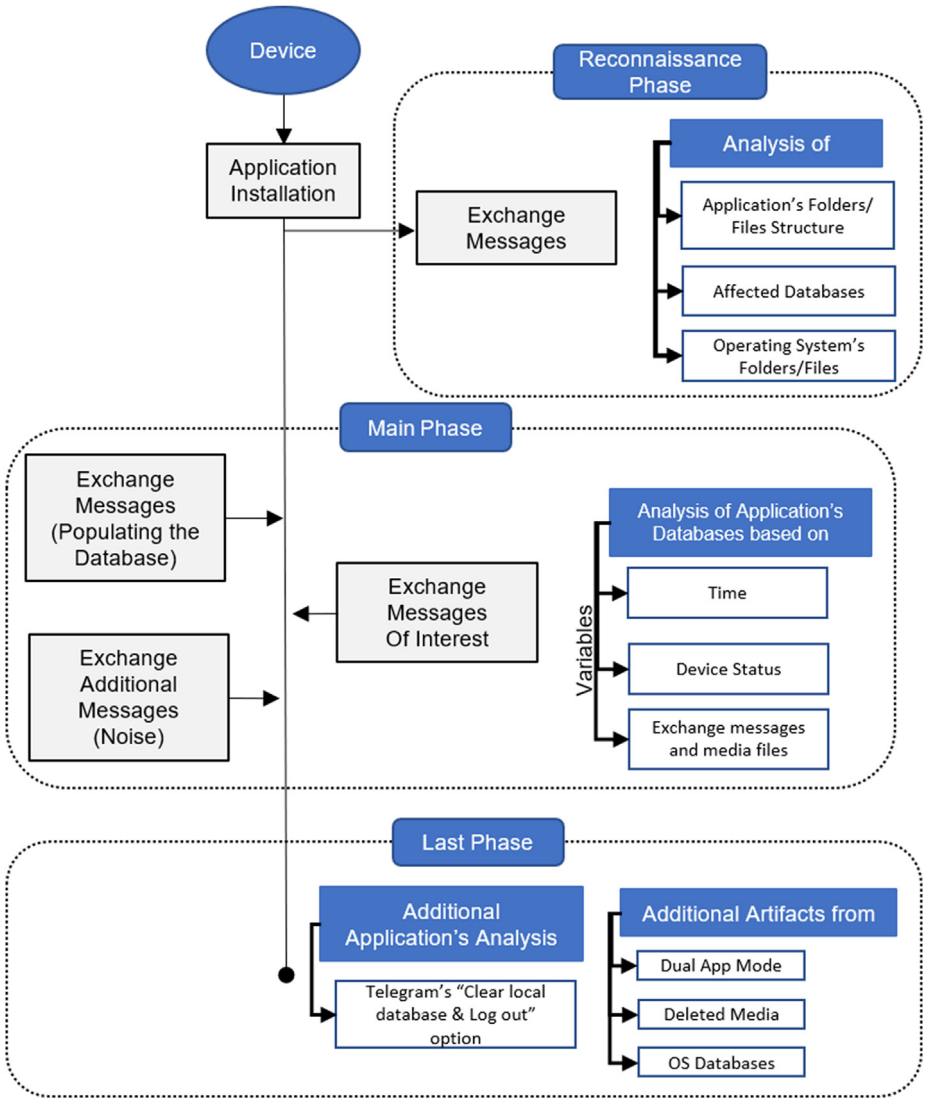

Fig. 1. Workflow of methodology.

Telegram's structure and how interacts with the Android OS, was made. In the second phase the volatility or the persistence of the deleted records was checked under three different variables/conditions: (1) the elapsed time until device's forensic acquisition, (2) the device's statuses (power on/off, airplane mode) and (3) the interaction with the messaging application itself (e.g., creating, reading, deleting messages). In the last phase, several Telegram's artifacts residing in the device examined, as well as the possibility of retrieving any information from its Servers.

Experiments Data: In order to examine as many potential occasions an examiner may face, as possible, several forensic images were acquired, in various time frames and in different statuses of the devices (Table 2). Time frames included the image acquisition of the device or the database of interest in specific time intervals. On the other hand, (1) leaving the device in normal mode (standby), (2) placing it in a specialized Faraday Box, (3) rebooting or shutting it down and (4) terminating all device communications by setting it in Airplane mode were examined as devices possible different. Total conducted experiments number is shown in Table 3.

During the experiments many media files were exchanged created both by default camera application and the Telegram's camera feature. All the media files were deleted manually and removed as well from the trash bin folder. For the extraction and analysis of the devices' and Telegram application's data a vast number of forensic tools commercial and non were utilized (Table 4) and compared. In order to monitor the alterations in any interaction with the application targeted acquisitions of specific directories or files (database) were made via pull commands.

By no means the utilized tools in this Paper are not the only ones that can be used for the same purposes.

Table 2
Reconnaissance phase variables.

Status	Time Frame	Action
Standby	As soon as possible	None
Standby	After 2/6/24 h	None
Standby	As soon as possible	Removing Sim
Airplane mode	After 2/6/24 h	None
After reboot	24 h	Removed Sim
After shutdown	48 h	None
Faraday box	After 2/6/24 h	None

Table 3
Total conducted experiments' number.

Physical Images	Acquiring Database's folder			Exchanged Messages
	Reconnaissance Phase	Main Phase	Last Phase	
12	6	180	–	2.200

Table 4
Tools utilized during the study.

Usage	Tool	License
Forensics Analysis	Access Data (Forensic ToolKit-FTK) 7.4	Commercial
	Autopsy Suite 4.19.1	Open Source
	Belkasoft X 1.10	Commercial
	Cellebrite UFED Physical Analyzer 7.50	Commercial
	Magnet Axiom 5.7, 6.0	Commercial
	MSAB (XRY and XAMN)	Commercial
	Oxygen forensics (Oxygen Detective) 14.1	Commercial
	slo-sleuth/android blob cache (https://github.com/slo-sleuth/android_blob_cache.) (media files)	Open Source
	X-Ways Forensics 20.1 (& HEX analysis)	Commercial
SQLite Analysis	aLEAPP 1.9.9	Open Source
	Andriller 3.6.1	Open Source
	Belkasoft X 1.10	Commercial
	bring2lite (Meng and Baier, 2019)	Open Source
	DB Browser for SQLite 3.12.2	Open Source
	Forensic Toolkit for SQLite	Commercial
	FQlite 1.5.5	Open Source
	KS DB Merge Tools for SQLite	Free Tool & Commercial
	Oxygen forensics (SQLite Viewer) 14.1	Commercial
	Paraben E3 Forensic Platform 3.1	Free Tool & Commercial
	SQLECmd (https://github.com/EricZimmerman/SQLECmd.)	Open Source
	SQLite-Deleted-Records-Parser (https://github.com/mdegrazia/SQLite-Deleted-Records-Parser.)	Open Source
	SQlite Expert 5.4	Free Tool & Commercial
	SQLite Forensic Explorer 2.0.0	Free Tool & Commercial
	Teleparser (https://github.com/RealityNet/teleparser.)	Open Source
	undark 0.6 (https://github.com/alitrack/undark.)	Open Source
Image Acquisition	ADB Commands (CLI) & SDK Platform Tools 31.03	Open Source
	Belkasoft X 1.10	Commercial
	Cellebrite UFED 4 PC 7.48	Commercial
	Magnet Acquire 2.45	Free Tool
	MSAB (XRY and XAMN)	Commercial
	Oxygen forensics (Oxygen Extractor) 14.1	Commercial

4. Telegram database's analysis and findings

Comparing the two examined mobile devices there were no differences between them, in the folder structure or the files of the application. In any case the initial size of **cache4.db** was 1.336 kb and its corresponding WAL file was 4.475 kb. The database was composed by fifty-two (52) tables. Forensically, the four most interesting tables for the examined scenario were the *dialogs*, *enc_chats*, *messages,* and *users* whose most valuable fields are presented in Table 5. The meaning of the fields is a result of this research combined with the one of Anglano et al. (2017). Moreover, tables *user_contacts_v7* and *user_phones_v7* contained information about the users participating in chat threads.

By executing queries or parsing with forensic tools (Table 6) combining the fields shown in Table 5 several conclusions were drawn from the analysis of the data inside **cache4.db**. An example of query results is presented in Fig. 2.

Some initial conclusions were:

- The same message resides in the WAL file several times in different states.
- In a secret chat the participants name could not be directly decoded, and a special process had to be done (Fig. 6).

Table 5
Total conducted experiments' number.

Field	Meaning
	Table: dialogs
did	Dialog id of the secret or normal chat
date	10-digit timestamp of a chat-thread's last message in Unix
unread count	Information about the unread messages of the dialog
last_mid inbox_max outbox_max	In conjunction could clarify the size of the dialog
	Table: users (All users, including owner)
uid	User's unique id number 10-digit form
name	User's name
data	User's connected number in binary form (BLOB)
	Table: enc_chats
uid	Secret Chat's User unique id number
user	User's unique id number (connected with uid of table users)
name	User's name (connected with name of table users)
	Table: messages
uid	Participant's unique id
read_state	Indicates whether a message has been read
send_state	Clarified whether an outgoing message has been sent
date	Timestamp of the message in Unix Seconds
body	Body of text messages, along with several other information, in a binary serialized form (BLOB)
out	Indicates a message's direction
TTL	Autodeleting messages timer (in seconds) set by the user