Retrieving deleted records from Telegram

Žan Počkar
zp68409@student.uni-lj.si
Faculty of Computer and Information Science,
University of Ljubljana
Večna pot 113
SI-1000 Ljubljana, Slovenia

Tom Sojer
ts22842@student.uni-lj.si
Faculty of Computer and Information Science,
University of Ljubljana
Večna pot 113
SI-1000 Ljubljana, Slovenia

ABSTRACT

Mobile applications utilize their own mechanism of storing data on a local mobile device. One interesting instant messaging platform that provides a mobile client is Telegram. In this paper, we focus on some of the research [5] done on the matter of how Telegram encodes and stores data. We present results of a forensic analysis made on two devices that used the application.

KEYWORDS

Mobile forensics, Telegram, SQLite, Write-Ahead log

1 INTRODUCTION

Telegram is a cloud based messaging service. It was first released for Android in 2010 and by now works on every major operating system. By far the most usage represents Android, as 85% of all the users use this OS. It supports different communication options, such as so-called normal chats, secret chats, video calling, chat rooms, etc. Normal chats do not support end-to-end encryption, but instead use an internal protocol MTProto that encrypts data on Telegram's servers. For end-to-end encryption, the secret chat method is mostly used.

As claimed by Telegram in June of 2022, they were one of the top 5 downloaded apps of that year with more than 700 million monthly users. This is why research is crucially needed to catch any potential flaw that could be exploited in a malicious manner.

In our article, the following will be presented: the background section (2), where we will go into details of Telegram Messenger and how the app stores data, then we will present some related work (section 3). The follows a review of how the authors of the discussed article prepared for the investigation (subsection 4.1) and analysis (subsection 4.2). We will show forensic tools useful for research (section 5) and discuss methods utilized in the reference article (section 6).

2 BACKGROUND

2.1 Database

To understand the behavior of the app's storage, we need to go into details about how it handles the database. The examined application version was 7.9.3, released in August, 2021. Telegram Messenger client stores data with SQLite. This database's method of storing is using a Write-Ahead log file (WAL) with a checkpoint of 1000 pages (a database in SQLite is divided into pages, each of size at least 512 bytes). Using the WAL file is a method of ensuring that data doesn't get lost. Before anything is written to a database, it is first logged. This ensures that any transaction can be repeated if some failure appears. The checkpoint of 1000 pages notes that all data before it reaches that point has been written to disk. In case of a crash, the recovery procedure would include finding the checkpoint and recovering anything until reaching the checkpoint. Meanwhile the main database is left intact and changed after that point has been reached. This means that recently deleted data can be found in the WAL file, but not in the main database file.

2.2 Telegram Data Structure

While Telegram stores some local data in clear text, such as the name of the sender/recipient, more volatile data is stored into complex data structures that appear in a binary serialized form. Fields in the structure are stored as a sequence of bytes, where they appear in specific positions. Retrieving information from such a structure demands first to deserialize it and then decode each useful field. The main problem is that only for a limited set of such structures, the structure and serialization scheme is known. The exact decoding scheme is not made public by the brand, which is why the process of figuring out the correct structure is left to the community. The structures are constantly redefined or removed after new features are released, which makes it harder to maintain tools that decode all the different variations.
Since a part of the application is open source, Anglano et al. [2] performed a source code analysis to identify all the different data structures used by the app as well as to reconstruct the structure and serialization schemes. The cited article named these structures *Telegram Data Structures* or *TDSs*. We will take a look at TDSs later in the analysis.

3 RELATED WORK

Although many different tools and solutions were used in this paper, individual use cases, justifications for the choices of tools and procedural workflows were not documented in detail. The related work section helps to explain some of those uncertainties as it lists works that depict guidelines, techniques, protocols, and standardized procedures in the field of mobile forensics, records retrieval, and data carving. They also credit works explaining (then relevant) Telegram's folder and database structure and the possible forensic analysis approaches, as well as works on similar messaging applications.

Some articles have already dealt with TDSs on other platforms. One instance is a paper, written by Gregoio et al. [3], which deals with and investigates the app on a Windows phone. The importance of forensic analysis has also been introduced on other messaging platforms such as WhatsApp [1] and WeChat [6].

DOI: https://doi.org/10.26493/scores23.03

As it was mentioned in the paper, with each version of Telegram released, the structure of its database, data structures, objects and, in general, the way data is handled and processed by the app, change as a result of added or removed features. This means that most, if not all, of the previously designed protocols and tools will have reduced success rates of analyzing the data. There has however been some previous research on the SQLite structures for data carving [4].

Despite the app's rapidly changing internal structure, older tools, documentation and works remain relevant due to the cross validation of the results which has proven itself to be essential. Using a plethora of tools and various versions of the same tools could yield significantly better results.

4 METHODOLOGY

4.1 Preparation and collection

To prepare for an examination and analysis and get a sense of how deleted records are handled by the application using different forensic tools, an investigative environment was first established. Two Android phones were examined: an LG G6 with Android 9 and Samsung A50 with Android 11. Both phones had a SIM card activated and Telegram Messenger client installed. Root access was gained on both devices in order to collect data used by the app. All data on the two phones that was unrelated to the observed app was deleted. This allowed the researchers to focus on relevant data only.

After finishing with the experimental setup, messages were exchanged between the clients. The preparation for the analysis included the exchange of media files, such as images and videos. All of the media were later deleted both from the app and the trash bin folder. In total, around 2200 messages were exchanged during this process [5].

The research focused on retrieving data in different scenarios. Criteria included (1) elapsed time since the device had been acquired, (2) whether the device is powered on, off or in airplane mode and the state of the application (3), i.e. if messages are created or deleted. As many different conditions were tested, several images of the devices were produced. For instance, in one of the images acquired, the phone was powered off and the researchers gained data from that state. By comparing different criteria, some potential real world scenarios were simulated.

4.2 Examination and analysis

As part of the forensic analysis, some open source and commercial forensic tools were used. This includes software for SQLite analysis and image acquisition software. We will discuss other tools later in the article.

There are a few forensically relevant data sources on the device. Firstly, the main local database, named cache4.db. In one of the scenarios, the size of cache4.db was 1336 kB, while the related WAL file had the size of 4475 kB. This is a consequence of using the Write-Ahead log. The local database included 52 tables, containing user data, messages, contacts and phone numbers. There is also a user configuration file that stores details of the account on the device, profile photos of user's contacts and also copies of files the user interacted with. The latter is stored in the media folder. From the normal chat messages table, several table entries could be read in clear text, such as the name of sender/receiver, date sent, the message status and direction. Info and body entries of this table were in the form of the Telegram Data Structure, which as previously defined, is in a binary form. Using forensic tools, some data was queried and the initial conclusions were:

- The same message may appear in the WAL file several times.
- The Auto-delete feature of Telegram, that removes messages after a certain period, did not affect the ability to recover messages.
- The body of messages is in a TDS form and not easily interpreted.

Experiments concluded that while different states of the device result in the cache4.db and WAL file changes, they do not alter the data inside of the messages table. For instance, when a phone has a network connection, both files might be altered after a short amount of time. They will remain in the same state if a phone is in airplane mode or the SIM card is removed. While cache4.db and the WAL file change very frequently, the messages tables inside doesn't. The researchers then observed events when the messages table was modified, and this happened when:

- a message was created, even if not successfully sent
- a message was received
- a message was opened
- a message was deleted

Hex editor was used to retrieve information. Now let's describe some other observations. Firstly, there was more storage needed to record an event when an outgoing message appeared compared to an incoming message.
Secondly, the main database file does not have the auto vacuum feature enabled which keeps the file size at the minimum. This potentially means that some of the deleted records might be retrieved. This was tested by the examiners on both devices. They deleted hundreds of records and attempted to recover them. Immediately after deletion, most of the messages were recoverable, but after new messages were created, all the deleted messages were not recoverable anymore. This was verified in the database file and the WAL file using a hex editor. Records are also not recoverable after the user has logged out.
Here, we can also make an observation with the WAL file. Since the file makes a commit to the database after a 1000 pages have been reached, a lot of data can be restored for a while if the commit doesn't occur for a longer period of time. In this sense, deleted records may be retrievable for a while, but here an element of luck is involved.
After deletion of media files that were used in any message exchange, some artifacts were still available on both the devices. This was observed by first deleting pictures and by comparing their artifacts with the records in cache4.db.

Lastly, we can ask ourselves and observe when records are not available anymore. This is the case on two occasions:

- when a user logs out and the database is deleted
- when the local database is deleted in the settings menu

5 TOOLS

Tools used for this research can be split into forensic and non-forensic tools. Non-forensic tools were used mainly for preparing the white box environment for the experiment, screen capturing and monitoring purposes. On the other hand, forensic tools were used mainly for extraction and analysis of both devices and Telegram app data.

5.1 Non-forensic tools

Among the non-forensic tools used for setting up the devices Magisk Manager was used for rooting and Team Win Recovery Project - TWRP for creating a custom recovery menu. In addition to the third-party solutions, an inbuilt android feature was used for creating additional user on each of the two devices. Dual apps or Dual messenger are features of contemporary Android devices which enable creating a virtual copy of an app that can then be used with a separate account as an independent user with their own file system, databases and permissions.

5.2 Forensic tools

A total of 26 tools were used, of which 13 were commercial use tools, 9 either open source or free and 4 were commercial tools with some form of a free version. As it was previously mentioned, the version of the tool used often significantly impacted the quality of the obtained results. Which is why in addition to exploring different software solutions, a combined use of multiple versions of the same tools was utilized.

All of the tools used were categorized according to their use case into forensic analysis, SQLite analysis and image acquisition tools. But not all tools were equally successful. Due to the rapid changes to the Telegram Data Structure, most tools were not up to date with the latest changes and could not decode any data. Some tools were successful, but not in all cases, some could successfully decode only the normal chat messages while some worked only with the secret messages. As the changes in the TDS are unlikely to backtrack, newer versions of forensic tools had higher success rates as they were more compatible with the latest TDS encoding. Examples of the above mentioned cases are Cellebrite UFED Physical Analyzer 7.50, Oxygen Forensics 14.1, and AXIOM v5.7 and v6.0.

Cellebrite UFED Physical Analyzer is a commercial tool used for uncovering digital evidence, trace events, and examine digital data. It allows for import and decoding of a specific application through its selective decoding, and it speeds up many aspects of the investigation including automatized report generation. Despite all the quality of life features, in this case, it was able to decode only normal messages.

Oxygen Forensics tools used in this research are marketed as all-in-one solutions for extraction, decoding, and analysis of both data and artifacts for both computer and mobile forensic investigations. Although convenient, these tools were able to decode only the secret messages.

When it came to AXIOM, built for remote acquisitions, collection and evidence analysis from different sources including mobile devices, the older version could not decode any data while the newer could decode only normal messages.

On the other hand, tools on which the researchers had relied the most were SQLite Analysis tools. Especially tools for inspecting, querying, carving and parsing the database. Although these tools also followed the trend of newer versions yielding better results.

6 DISCUSSION

This attempt to retrieve the telegram data can't be considered as nothing but a proof of concept and even that done in extreme laboratory conditions. While the article uses some novel approaches to retrieve stored data and does achieve some success. We must raise some questions about the methodology and the fine result of retrieving the data. We can generally split our questions into two categories. The first categories deal with the methodology of the used device and acquired data. The second category deals with the final results and readability of said data.

We first must raise questions about the devices used. The team decided to use two completely wiped devices on which the first installed a custom recovery menu and gained root access. Secondly, they installed the same version of the telegram app, a version which is now discontinued. Lastly, while the team did simulate conversation with the telegram app no other application was running on the devices prior, during or after communications.

It is true that the article itself freely admits that the device setup wasn't forensically sound but more of a whitebox proof of concept. And yet even after this question arise if the test wasn't done in too many perfect conditions and if such results even carry any weight in real world practice.

Let's begin with the devices themselves. The rationale for not using other applications to reduce during the test is sound but in our opinion flawed. Multiple times the data was retrieved only because the team was able to identify changes after the messages were exchanged, in addition the article states that the data is in certain instances volatile and can be lost if the changes occur on the operating device. It is true the team did test three different changes such as turning the device on and off that did not change the data. But the same test did show that actions inside the applications lead to such change. The question arises if the running of similar applications at the same time could lead to trigger such changes. In this vein of questioning is the decision to use just a limited amount of accounts while we understand that because of limited time and money for any such research smaller sample sizes must be used. During the research, the team has shown that a message from any source using the telegram application will modify the already stored data. That means that an average user's message would be much harder to retrieve especially if the user is an active user of the application.

While the usage of the same version of the application is ideal, this is a likely event for users which use the application regularly so we don't see much problem with this approach even if it is a little idealistic. Another problem with the usage of this version of application is the fast change rate for the application. As many application telegram changes quickly and frequently this means that approaches that work on one version may not work on newer versions.

And lastly gaining root access in advance and installing custom recovery software before installing the telegram application is a

problem in the methodology. As we can't be sure how gaining root access or installing the recovery menu, could change the data we are trying to access but this could be mitigated through a usage of multiple copies used in any forensic investigation even if it is extremely time consuming.

Gaining root access may also raise extra problems regarding who and why it is being done. Gaining access during a lawful police investigation with proper warrants is no concern, but gaining root access by a private individual for other purposes may be illegal and subject to punitive actions from local authorities.Meaning that any method that requires such access can't always be used reliably in all circumstances.

The other major category of the question raised is pertaining to the results themselves. They have had moderate success in retrieving different data from the telegram application, they freely admit that they have much greater success with text messages in comparison to the pictures. But the thing one must question is if the success from retrieving text data shouldn't be classified more as a partial or limited success. The problem which is already mentioned in the articles is the format of encoding used in telegram text data. While the team was on multiple occasions able to retrieve the raw data they couldn't read large or even in some cases whole portions of retrieved data as data was parsed inside the TDSs structure. One must then ask themselves if retrieved data that can't be decoded in reasonable timeframes can't even be considered retrieved or must it at most be just used as proof of communication between parties. The whole question of usability of the retrieved data does not only rests with the team conducting the research as the research has shown that most modern and available current forensic tools aren't capable of decrypting the data. In the future, if better tools for decrypting the retrieved data are developed, such a method for retrieval may be used for more than just proof of communication.

7 CONCLUSION

We find that the article has a straight forward goal, clear methodology and a concise plan for proving its findings. In the article, the team thoroughly presents the Telegram applications, the methodology they used, their process and their results.

Conclusions regarding volatility of cache4.db database and the Write-Ahead log file along with the data collected from other articles on Telegram could be used in further investigations. In conclusion, we believe this article is an important first step at looking into retrieving the deleted records from the Telegram Messenger client. Further analysis would be needed in regards to discovering possible vulnerabilities of the application as too many tradeoffs and ideal conditions were used in the primary research for this article.

REFERENCES

[1] Cosimo Anglano. 2014. Forensic analysis of WhatsApp Messenger on Android smartphones. https://www.sciencedirect.com/science/article/pii/S1742287614000437. *Digital Investigation* 11, 3 (2014), 201–213. Special Issue: Embedded Forensics.

[2] Cosimo Anglano, Massimo Canonico, and Marco Guazzone. 2017. Forensic analysis of Telegram Messenger on Android smartphones. https://www.sciencedirect.com/science/article/pii/S1742287617301767. *Digital Investigation* 23 (2017), 31–49.

[3] J. Gregorio, A. Gardel, and B. Alarcos. 2017. Forensic analysis of Telegram Messenger for Windows Phone. https://www.sciencedirect.com/science/article/pii/S1742287617301032. *Digital Investigation* 22 (2017), 88–106.

[4] Dirk Pawlaszczyk and Christian Hummert. 2021. Making the Invisible Visible – Techniques for Recovering Deleted SQLite Data Records. https://conceptechint.net/index.php/CFATI/article/view/17. *International JOURNAL of Cyber Forensics and Advanced Threat Investigations* 1, 1-3 (2021).

[5] Alexandros Vasilaras, Donatos Dosis, Michael Kotsis, and Panagiotis Rizomiliotis. 2022. Retrieving deleted records from Telegram. https://www.sciencedirect.com/science/article/pii/S2666281722001287. *Forensic Science International: Digital Investigation* 43 (2022), 301447.

[6] Songyang Wu, Yong Zhang, Xupeng Wang, Xiong Xiong, and Lin Du. 2017. Forensic analysis of WeChat on Android smartphones. https://www.sciencedirect.com/science/article/pii/S1742287616301220. *Digital Investigation* 21 (2017), 3–10.