Digital Forensic Analysis of Telegram Messenger on Android Devices

Conference Paper · October 2016

DOI: 10.1109/ICTS.2016.7910263

CITATIONS
4

READS
6,699

3 authors:

Gandeva Bayu Satrya
Telkom University
25 PUBLICATIONS **91** CITATIONS
SEE PROFILE

Philip Tobianto Daely
Kumoh National Institute of Technology
18 PUBLICATIONS **71** CITATIONS
SEE PROFILE

Muhammad Arief
Telkom University
9 PUBLICATIONS **17** CITATIONS
SEE PROFILE

Digital Forensic Analysis of Telegram Messenger on Android Devices

Gandeva Bayu Satrya
Telkom Applied Science School
Telkom University
gbs@telkomuniversity.ac.id

Philip Tobianto Daely
IT Convergence Engineering
Kumoh National Institute of Technology
20156130@kumoh.ac.kr

Muhammad Arif Nugroho
Telkom School of Computing
Telkom University
arif.nugroho@telkomuniversity.ac.id

***Abstract*—We provide a thorough description of all the artifacts that are generated by the messenger application Telegram on Android OS. We also provide interpretation of messages that are generated and how they relate to one another. Based on the results of digital forensics investigation and analysis in this paper, an analyst/investigator will be able to read, reconstruct and provide chronological explanations of messages which are generated by the user. Using three different smartphone device vendors and Android OS versions as the objects of our experiments, we conducted tests in a forensically sound manner.**

***Index Terms*—Android forensics, Telegram, remnant data.**

I. Introduction

Short Message Service (SMS) was a widely used service in the 2G and 3G eras. However in the 4G era things have changed; SMS is being abandoned, and there is a significant increase in the number of social media and social messenger applications. Social media applications and social messenger applications are online media that can be used by users for free, to share and exchange information in the form of texts, images, and videos. Various social messenger applications offer a wide variety of interesting features such as text chat, group chat, message notifications, share location, share contacts, status updates, and file sharing.

Based on data from November 30^{th} 2015, the world's Internet users reached 3,366 billion with Asia in the highest position [1]. A wide range of applications can be displayed and accessed in accordance with the users wishes. This perceived ease of internet access can lead to unlawful acts. Law violation which utilizes computer technology with Internet access is hereafter identified as cybercrime [2][3][4]. Identity theft, transaction fraud, blackmailing, and other types of criminal acts using computers/smartphones are all considered as cybercrimes.

This paper describes an investigation into digital forensics on social messenger applications on Android smartphones [5]. These various types of messenger applications are available in Google PlayStore and can be downloaded for free. There increased use makes understanding these applications important. It is expected that this paper could help forensics investigators / analysts in facing cybercrime cases which utilize social messenger applications. The case study used here is of the application Telegram. The contributions of this paper can be summarized as follows:

1) We discuss the process of acquisition, analysis, and interpretation of metadata obtained from Telegram.
2) Furthermore we show how the remnant data relate to each other within the process from the acquisition to the analysis, such as when the user carries out the installation process, signup / in, add / delete / block contact, message sending process (text, figures, or voice), location/file sharing, sign out, and uninstall.

In section II we review existing work. In section III we explain the methodology used in the analysis process. Then in section IV, we explain in detail the process of forensics analysis conducted in this paper. Finally, in section V, we give the conclusions of this paper.

II. Related Works

Husein et al. 2009 [6] studied and reported the forensic analysis of three different instant messaging applications (IMs): AIM, Yahoo! Messenger and Google Talk, (both client based and web based versions) showing that various useful artifacts related to IMs can be recovered from the iPhone, including username, password, buddy list, last log-in time, and conversation timestamp as well as conversation details. Forensic examination of Instant Messaging on smart phones such as the iPhone pose a new challenge for investigators as well as researchers due to the uniqueness of the file system.

Mahajan et al. 2013 [7] conducted forensic data analysis of two widely used IMs on Android phones: WhatsApp and Viber. The tests and analysis were performed with the aim of determining what data and information can be found on the devices internal memory for instant messengers e.g. chat messaging logs and history, send & received images or video files, etc. But this paper does not cover the details about the found traces and evidence as it is limited to normal chat scenario only.

Anglano [8] was able to reconstruct the list of contacts and the chronology of messages that had been exchanged by users. The correlation of the contents of the chat database with the information stored in the log files allows the investigator to determine which messages have been deleted, when these messages were exchanged, and the users that exchanged them. But the usage of any hash function is not performed in this paper.

978-1-5090-1381-4/16/$31.00 ©2016 IEEE

In this paper, we conducted Android forensics on the internet messenger application Telegram. Three smartphones, Xiaomi Redmi Note 1W Android version 4.4.2, Samsung Galaxy Note II SHV-E250K Android version 4.4.2, and the Samsung Galaxy S4 GT-i9500 Android version 4.3 were used. By consistently using forensically sound principles, we conducted acquisitions and analysis. The difference between this paper and the previous one is that an offline mobile forensics process was performed on user activities (install, login, insert / update / delete contact, chat text, multimedia chat, and location/file share), and log live mobile forensics. To strengthen digital evidence legitimation in court, SHA-1 was used as a hash value.

III. Analysis Methodology

A. Identifying

Identifying is deciding which materials to use as digital evidence, where said evidence is stored, and assessing the impact of the activities that will be performed by the user [9]. The role of the writer in this case is as a forensics investigator / analyst. For this paper three smartphones were used as objects of experimentation. These were Xiaomi Redmi Note 3G, Samsung Galaxy Note II LTE, and Samsung Galaxy S4. The social messenger application used as a case study was Telegram version 3.4.2 for Android OS.

B. Preserving

This is the most vital process of digital forensics [9]. To uphold the values of cyberlaw, it is of utmost importance to record events or steps which will be performed on electronic evidence. It is an absolute requirement that an investigator has legitimate and justified experience or certification. It is also of utmost importance that the research is conducted in a forensically sound manner in order to minimize the occurrence of data changes, and if changes do occur, it must be explained and accounted for before the courts.

The acquisition process can only be conducted on a rooted smartphone. The acquisition process in this research, used Android Debug Bridge tool v1.0.32, SQLite Browser v3.8.0, Busybox v4.1, Root Browser v2.2.3, Online Nandroid Backup v4.4.5, dex2jar v2.0, sha1sum (Linux Ubuntu 14.04) and Notepad++ v6.8.8.

C. Analyzing

Extraction, processing, and interpretation of digital evidence is the most important element in the forensics analysis [9]. From the digital evidence produced in the preservation process, data cannot be directly read or directly taken. With the help of tools that were mentioned before, we were able to proceed to the analysis process. The analytical method we used was to compare the database and directories before user activities and after. If the hash value obtained changed then deeper analysis of that file was performed. In addition to offline forensics, we also used a live forensics log as supplementary metadata / remnant data. Analysis and testing was divided into five categories:

1) Application and user activity: install, signup, and sign in
2) Contact information: adding, update, delete, blocking contact
3) Messages exchanged: text (one to one and one-to-group), multimedia (one to one and one-to-group), and voice message
4) Sharing: file and location
5) Deleted communication: clear chat, sign out, and uninstall

D. Presenting

The main objective of digital forensics is to help prove cybercrime in court [9]. Here we give the results from all the activities of the mobile forensics that were performed. Hopefully this guidance can be useful to researchers, investigators and forensics analysts, and cyberlaw practitioners.

IV. Telegram Digital Forensics

A. Research Preparation

The scope of this research was to determine the remnant data in Android devices from the utilization of Telegram messenger activities, such as: installation, sign up, sign in, contact (add, update, delete, block), text chat (one to one communication and one-to-group), multimedia chat (one to one communication and one-to-group), sharing (file, contact, location), history (clear history and delete chat), log out, and uninstall. The first test took place on the Samsung Galaxy Note II for a total of seventeen testing activities. Then, Xiaomi Redmi Note 3G was used as a user-1 message sender / receiver. The Samsung Galaxy S4 was used as a user-2 sender / receiver which was useful for further testing related to communication group.

To find the remnant data we used offline forensics (from the backup file smartphones) and online forensics (from live whilst the activities took place). Then to make sure that obtained digital evidence would be accountable in court, in each of these activities the hash value was taken using SHA-1 check sum [4]. Then at the analysis step, the data that was acquired was compared before and after each user activity. If a different hash value was found, only then could analysis be conducted. Then at the end the remnant data that is generated by Telegram Messenger on Android devices is presented, including file names, locations and contents.

B. Post-mortem Investigation

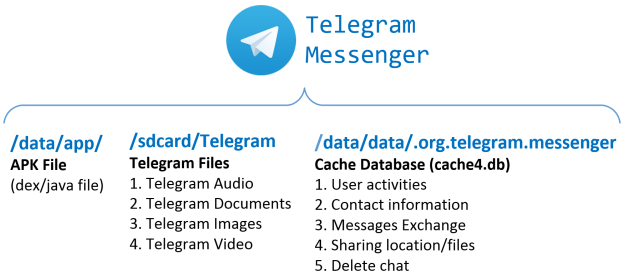

Fig. 1: Structure Telegram Forensics Analysis

Fig 1 shows the results of manual digging in this research. Broadly speaking, Telegram Messenger has three important files which are Telegram.apk, Directory storage files located in the local memory of the smartphone, and databases of all the activities to be carried out in further testing and analysis. To simplify the analysis, the authors divided this scenario into five broad scenarios branched out into more specific sub scenarios. The scenarios are application and user activity, contact information, messages Exchange, sharing, and deleted communication. In each sub-section there is a detailed description of any activities that were performed by Telegram users.

1) Application and user activity: The smartphone was in a rooted state, in which all previously mentioned tools had been successfully installed. The chain of activities for this research was first opening the Google Play Store, conducting the installation process, then proceeding with the sign up and sign in. Two methods were used, namely online and offline forensics. In Table I are the results of the online forensics that were obtained *adb logcat*. Timestamp of obtained logcat could be used as digital evidence in court later.

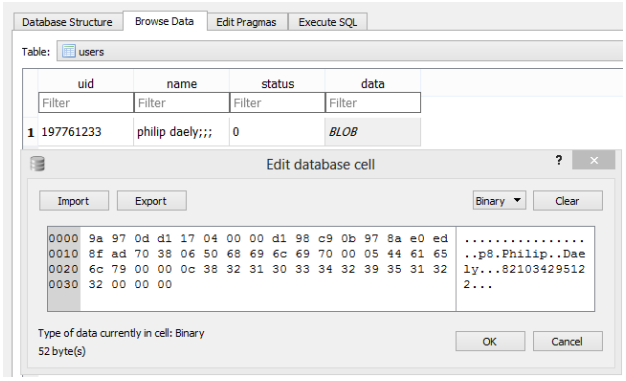

Fig. 2: Evidence user already signup

For the investigator to obtain user information regarding who logged in along with the telephone number used for the registration process, offline forensics is required. Offline forensics was obtained using *adb backup -all* command. Then it was analyzed with the help of Linux. The reason for using Linux was for finding which files were changed and to make the sorting process easier. As mentioned in section Analysis Methodology we used the method of comparing hash values prior and post activity. With this method, we got information on which files were changed. After that analysis was performed on the database of said changed file.

For the sign up activity the registered phone number is found in the database by performing analysis directly on `\data\data\org.telegram.messenger\files\cache4.db` directory on the `users` table. As shown in Fig 2, it can be seen clearly (without encryption), the association between users who sign up and the phone number that was already registered with the format *userID, full name with nickname,* and a *operator phone number* that was already registered. Furthermore in its database, Telegram used BLOB to filter all data. As is written on the Robo Realm website the purpose of using BLOB filters is that it is reliable and dependable despite noisy background [10]. We did not discuss filtering, because the focus of this research was the analysis of mobile digital forensics.

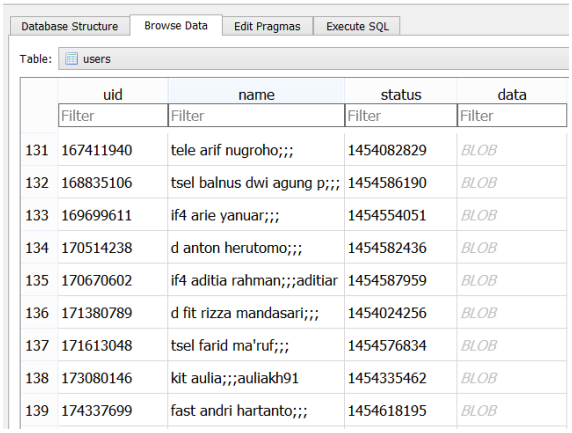

Fig. 3: Evidence database users in table users

2) Contact information: The same method was used to obtain metadata associated with contact information such as: adding contacts, update contact information, delete, and blocking contact. An online forensics log using logcat got the information related to these activities as shown in Table II. It could be used as a reference for investigators / analysts in the search for digital evidence and in strengthening the offline analysis.

As for offline forensics, the files are found in the same database `cache4.db` but are displayed in different tables. For contact information, Telegram stores in the `user` table as in Fig 3 above. In the user table userID, name (with nickname), status (the last seen status), and data (SQLite will show the full name and number of the service used by the user) will be seen. User information not only exists on `users` table, but also in tables of `user_contacts_v6` (will look userID, forename, and surname) and `user_phones_v6` (will show userID, phone number, and deleted).

The delete user scenario was correlated with `user_phones_v6` tables. In the delete column is shown `1` which means it has been deleted, or otherwise `0` if the user is still stored in contacts. For the blocking contact scenario, in which before the activity the `blocked_users` table was empty, after block has been conducted against the user (uid: `173080146`), table `blocked_users` on `cache4.db` the database is filled with the user id.

3) Messages exchange: In this scenario, there were three sub-scenarios which were chat text, multimedia chat, and voice messages. By using the *adb logcat*, information regarding the chat text and multimedia along with the time and date of the incident was obtained as displayed in Table III. Everything is clearly visible for each activity that was used including matching timestamp against `cache4.db` database. The total number of logs is massive, and thus we just take the most important ones.

If offline database forensics on `cache4.db` is conducted, then on the table `messages`, there will be visible all the information from particular users about chats that have been done both on one-to-one or one-to-group communication. Fig 4 is an example of the output text message sent from userID `179433774` to groupID `58700220` along with the time it happened. Subsequent analysis that can be obtained from here is the text that is displayed in this table. This is unencrypted in plain text. Thus, the investigator/analyst can search for remnant data information related to chats that may have been used in cybercrime or cyberterrorism.

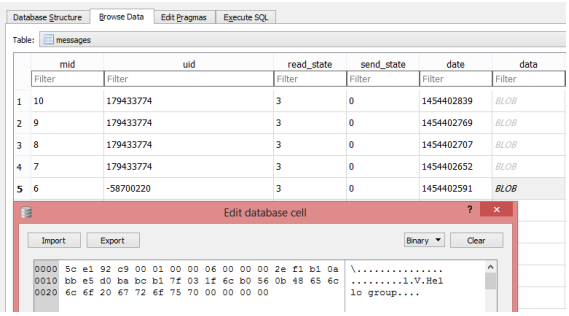

Fig. 4: Evidence database messages in table messages

In addition to text chat, the authors also show evidence related to multimedia chat conducted by the sender (uid: `179433774`) to the recipient (uid: `197761233`). In Fig 5 it can be seen that the sender has sent the image named (along with its directory) `/storage/emulated/0/Android/data/org.telegram.messenger/cache/2147483648_210001.jpg` and that it was copied to the `/sdcard/Telegram/TelegramImages/2147483648_210001.jpg`.

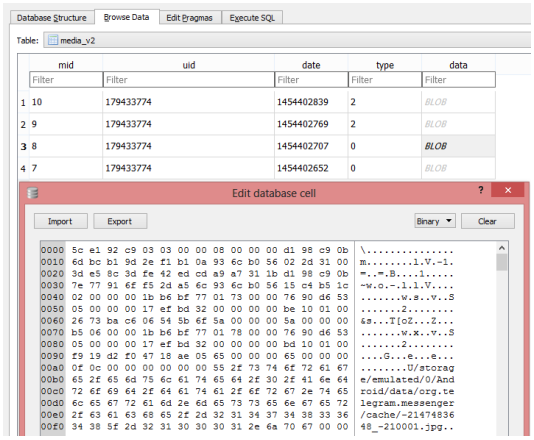

Fig. 5: Evidence multimedia files during chating

Fig 6 shows digital evidence related to groups. It shows remnant data information with the format of groupID (uid `58700220`), group name (WENS Forensics), and data (metadata format). If we look closer we see that the UID has ten digits and the group ID has eight digits. However, the number of digits will increase according to the number of users enrolled in the Telegram server membership. The next analysis is uid layout for groups, shown in the table `chats`. Uid for users is shown in the table `users` of the database `cache4.db`. Here we can also see the uid analysis for Telegram itself which was used to broadcast messages from a central server. Telegram itself uses uid `777000` or a 6 digit number. All chat data can be found on the messages table. The main difference between one to one and group chat is that one to one is only visible in common numbers (10 digits) while the group chat has ”-” signs next to it, as seen in Fig 4. Uid with the code ” `-58700220`” is for the group, while uid with code `179433774` is for a single user (sender).

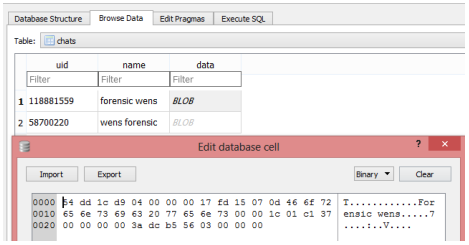

Fig. 6: Evidence group name that already join

For the voice message scenario, we can obtain voice-related files that are sent from the sender to the receiver and vice versa. From Fig 7 it can be identified that the `Telegram Audio` folder contains several files that are related to voice chat with _name.ogg file name format. If we have an audio player that supports ogg files, it can be opened directly. If not, we can use a converter such as, *ogg to wav*, in which it will produce a file with the same file name but different extensions such as `5_770662191127855105.wav`. Then it can be played in Windows Media Player with the same sound quality and without encryption.

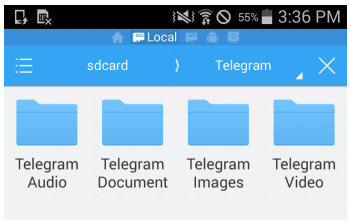

Fig. 7: Stored files by Telegram in local memory

4) Sharing: We divide this scenario into two sub-scenarios which are file sharing (sender sends a file with the extension PDF) and location sharing (sender sends the position they are located). For the analysis of the live forensics log, using ADB Logcat, a timestamp to the event is shown in Table IV. As for offline forensics we used a file that had been acquired and the same analysis was done as for previous activities. Later on the files could be found in the shared `Telegram Documents` folder as already shown in Fig 7. In this case the sharing location was stored in addition to a file in jpeg format. If the image is opened it shows a map with the persons location indicated.

5) Deleted communication: This scenario is divided into three sub-scenarios which are clear chat, logout, and uninstall. If clear chat is done on a particular user-1 then blob value for said user is null, but said user-1 is still displayed in the messages table. If delete chat is done on user-1, then mid and uid from said deleted user will be gone from the message table. Moreover, if there is a file that has been sent, it will also erase the data alias within the `Telegram Documents` folder (example). As for the logout scenario, the database file `cache4.db` still exists, but all the values of the table will be empty.

C. Discussion

This analysis provides a guide that can be used by digital forensics investigators, forensics analysts, and that should improve knowledge for cyberlaw practitioners in finding remnant data on the application Telegram Messenger. In Table V is a summary of all the results of post-mortem investigations analysis detailed above. 1-1 is one to one communication, while the 1-n is one to group communication. Seventeen scenarios were investigated. Numbers 1-14 were very easy to obtain, whereas for numbers 15-17 log forensics and offline forensics had to be used simultaneously. With log live forensics, the exact timing of every activities were able to be obtained. This investigation was performed with assumption that the smartphones have not been turned off from the time of activity happening until the time of investigation.

V. Conclusion

Based on the post-mortem investigation that has been done in five major scenarios (divided into seventeen smaller scenarios) on Telegram Messenger 3.4.2 application on Android devices, this paper details remnant data that could be used as digital evidence in cybercrime cases. In some scenarios we also show the interpretation of existing databases on contact and chat, both text and multimedia. The foremost conclusion is that this research can be used as a reference for forensics investigators and analysts relating to remnant data searches in cybercrime cases. There are many other Internet messenger applications that could be used in future case studies, as well as platforms other than Android and utilizing different vendors.

Acknowledgment

This work was supported by Laharisi. This work was supported by Forensics and Security Laboratory, Informatics Engineering Telkom School of Computing, Telkom University, Bandung, Indonesia. And also thank you for WENS Laboaratory, Kumoh National Institute of Technology, Gumi, South Korea.

References

[1] Internet World Stats. World internet usage and population statistics november 30, 2015 - update. http://www.internetworldstats.com/stats.htm. Accessed: 2016-01-01.

[2] G.B. Satrya and S.Y. Shin. Optimizing rule on open source firewall using content and pcre combination. *Journal of Advances in Computer Networks*, 3(3):308–314, 2015.

[3] Gandeva B. Satrya, Niken D.W. Cahyani, and Ritchie F. Andreta. The detection of 8 type malware botnet using hybrid malware analysis in executable file windows operating systems. In *Proceedings of the 17th International Conference on Electronic Commerce 2015*, ICEC '15, pages 5:1–5:4, New York, NY, USA, 2015. ACM.

[4] G. B. Satrya, P. T. Daely, and S. Y. Shin. Android forensics analysis: Private chat on social messenger. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 430–435, July 2016.

[5] Andrew Hoog. *Android forensics: investigation, analysis and mobile security for Google Android*. Elsevier, 2011.

[6] Mohammad Iftekhar Husain and Ramalingam Sridhar. iforensics: forensic analysis of instant messaging on smartphones. In *Digital forensics and cyber crime*, pages 9–18. Springer, 2009.

[7] Aditya Mahajan, MS Dahiya, and HP Sanghvi. Forensic analysis of instant messenger applications on android devices. *arXiv preprint arXiv:1304.4915*, 2013.

[8] Cosimo Anglano. Forensic analysis of whatsapp messenger on android smartphones. *Digital Investigation*, 11(3):201 – 213, 2014. Special Issue: Embedded Forensics.

[9] Rodney McKemmish. *What is forensic computing?* Australian Institute of Criminology Canberra, 1999.

[10] Robo Realm. Blob filter. http://www.roborealm.com/help/Blob_Filter.php. Accessed: 2016-01-01.

TABLE I: Installation and sign up log activity

Activity	logcat timestamp
Installing	02-02 14:28:54.128 4583 4583 D Finsky : [1] InstallerTask.advanceState: Begin install of org.telegram.messenger
Launch	02-02 14:29:23.058 4583 4583 D Finsky : [1] ReferrerRebroadcaster.onPackageFirstLaunch: Package first launch for org.telegram.messenger
Sign up	02-02 14:29:33.495 6108 6108 D HockeyApp: Looking for exceptions in: /data/data/org.telegram.messenger/files
	02-02 14:29:33.720 2504 2719 D ActivityManager: bindService callerProcessName:android, calleePkgName: org.telegram.messenger, action: android.content.SyncAdapter
SMS Verification	02-02 14:29:37.188 2503 2727 I TIMA: tlc_communication: Send Trustlet TCI Message
	02-02 14:29:37.188 2503 2727 I TIMA: tlc_communication: mcNotify is completed
Opening	02-02 14:29:38.443 2503 3069 D ActivityManager: startService callerProcessName:org.telegram.messenger, calleePkgName: org.telegram.messenger

TABLE II: Contact information log activity

Activity	logcat timestamp
Adding	02-02 17:37:34.049 4383 5006 D LPE_ANALYZER:InactiveAppAnalyzer: $a\$c : 566 >$ AppUsageStatsVO [id=null, appInfo=AppInfoVO [applicationLabel=Telegram, packageName=org.telegram.messenger, isUserApplication=true, hasLaunchIntent=true, lastAppExecTime=Thu Jan 01 09:00:00 GMT+09:00 1970, lastServiceRunningTime=Thu Jan 01 09:00:00 GMT+09:00 1970, installedTime=Fri Aug 01 21:00:00 GMT+09:00 2008, numOfExecution=0, totalRunningTime=0, numOfNoti=0, avgMemInService=0, startGatheringData=null, endGatheringData=null], date=Tue Feb 02 00:00:00 GMT+09:00 2016, startDate=Mon Feb 01 15:33:02 GMT+09:00 2016, lastUsedTime=Tue Feb 02 11:30:37 GMT+09:00 2016, runningDays=2, pointForInactive=1.0, startGatheringData=null, endGatheringData=null]
Updating	02-02 17:37:47.194 2504 2851 D ActivityManager: startService callerProcessName:org.telegram.messenger, calleePkgName: org.telegram.messenger
Deleting	02-02 18:29:33.495 6108 6108 D HockeyApp: Looking for exceptions in: /data/data/org.telegram.messenger/files
Blocking	02-02 18:30:07.435 2489 2489 D PopupWindow: invokePopup: mPopupView = org.telegram.ui.ActionBar. ActionBarPopupWindow$ActionBarPopupWindowLayout43540fa8 VFED.... .....I. 0,0-0,0, WindowManager.LayoutParams = WM.LayoutParams(296,50)(wrapxwrap) gr=#10000033 ty=1000 fl=#860000 extfl=#0 fmt=-3 flagsEx=10007f

TABLE III: Chatting log activity

Activity	logcat timestamp
Chat (S) Text	02-06 19:34:12.246 6089 6089 I dalvikvm: Could not find method org.telegram.ui.Cells.ChatBaseCell.onProvideStructure, referenced from method org.telegram.ui.Cells.ChatMessageCell.onProvideStructure
Chat (R) Text	02-06 19:34:12.311 6089 6089 I dalvikvm: Could not find method android.app.Activity.checkSelfPermission, referenced from method org.telegram.ui.Components.ChatActivityEnterView$8.onTouch
Chat (S) Image	02-06 19:36:01.221 6089 6089 I dalvikvm: Could not find method android.content.Context.checkSelfPermission, referenced from method org.telegram.ui.Components.ChatAttachView.updatePhotosButton
Chat (R) Image	02-06 19:37:07.241 6089 6089 I dalvikvm: Could not find method android.app.Activity.checkSelfPermission, referenced from method org.telegram.ui.PhotoViewer$2.onItemClick
Voice 1	02-06 19:37:38.996 6089 6089 V Vibrator: Called vibrate(long) API - PUID: 10212, PackageName: org.telegram.messenger
Voice 2	02-06 19:37:38.996 6089 6089 V Vibrator: vibrate - PUID: 10212, PackageName: org.telegram.messenger, ms: 20, mag: -1
Voice 3	02-06 19:37:39.001 2504 3089 D VibratorService: ImmVibe vibratorOff()

TABLE IV: Share location log activity

Activity	logcat timestamp
share location	02-06 19:41:05.935 2446 2446 W dalvikvm: VFY: unable to resolve new-instance 21 (Landroid/animation/StateListAnimator;) in Lorg/telegram/ui/LocationActivity;
	02-06 19:41:05.940 2446 2446 W dalvikvm: VFY: unable to resolve virtual method 61: Landroid/app/Activity;.checkSelfPermission (Ljava/lang/String;)I
	02-06 19:41:05.943 2446 2494 D skia : jpeg_decoder mode 1, config 6, w 296, h 526, sample 1, bsLength 906e!!
share file	02-06 20:14:36.968 2446 2446 D ListView: mSelectorRect.setEmpty in layoutChildren this=android.widget.ListView42697e58 VFED..CL .....ID 0,0-720,1118 #7f0c000d app:id/listView
	02-06 20:14:37.148 2446 2446 D ListView: mSelectorRect.setEmpty in layoutChildren this=android.widget.ListView42697e58 VFED..CL .F....ID 0,0-720,1118 #7f0c000d app:id/listView

TABLE V: Guidance for finding remnant data on Telegram Messenger

No	Activity	Location	Method
1	Installation	\data\data\app\org.telegram.messenger-1.apk	Offline & Online Forensics
2	Sign up / login	\data\data\org.telegram.messenger\files\cache4.db	Offline & Online Forensics
3	Adding contact	\data\data\org.telegram.messenger\files\cache4.db	Offline & Online Forensics
4	Update contact	\data\data\org.telegram.messenger\files\cache4.db	Offline & Online Forensics
5	Deleting contact	\data\data\org.telegram.messenger\files\cache4.db	Offline & Online Forensics
6	Blocking contact	\data\data\org.telegram.messenger\files\cache4.db	Offline & Online Forensics
7	1-1 Comm. Text	\data\data\org.telegram.messenger\files\cache4.db	Offline & Online Forensics
8	1-n Comm. Text	\data\data\org.telegram.messenger\files\cache4.db	Offline & Online Forensics
9	1-1 Comm. Multimedia	\sdcard\Telegram\TelegramImages*.jpg	Offline & Online Forensics
10	1-1 Comm. Multimedia	\sdcard\Telegram\TelegramImages*.jpg	Offline & Online Forensics
11	1-1 Voice Comm.	\sdcard\Telegram\TelegramAudio*.ogg	Offline & Online Forensics
12	1-1 Voice Comm.	\sdcard\Telegram\TelegramAudio*.ogg	Offline & Online Forensics
13	share file	\sdcard\Telegram\TelegramDocuments*[pdf|doc|zip]	Offline & Online Forensics
14	share location	\sdcard\Telegram\TelegramImages*.jpg	Offline & Online Forensics
15	clear chat	\data\data\org.telegram.messenger\files\cache4.db	Offline & Online Forensics
16	log out	cache4.db database will becoming NULL	Offline & Online Forensics
17	uninstall	Only directory \sdcard\Telegram* is exist	Offline & Online Forensics