**Mobile and Forensics (MF)**
Vol. 4, No. 1, February 2022, pp. 31-43
P-ISSN: 2656-6257, E-ISSN: 2714-6685, DOI: http://dx.doi.org/10.12928/mf.v4i1.5537

# DIGITAL FORENSIC ANALYSIS OF TELEGRAM MESSENGER APP IN ANDROID VIRTUAL ENVIRONMENT

**[1]Ahmed Raza, [2]Muhammad Bilal Hassan**
[1,2] School of Electrical Engineering and Computer Sciences (SEECS),
National University of Sciences and Technology (NUST), Islamabad, Pakistan
[1] araza.msis20seecs@seecs.edu.pk; [2] mhassan.msis20seecs@seecs.edu.pk*correspondence email

**Abstract**

*The paper provides an in-depth analysis of the artifacts generated by the Telegram Messenger application on Android OS which provides secure communications between individuals, groups, and channels. Since the past few years, the application went through major changes and updates and the latest version's artifacts varied from the previous ones. Our methodology is based on the set of experiments designed to generate the artifacts from various use cases on the virtualized environment. The acquired artifacts such as messages, their location, and data structure how they relate to one another were studied and were then compared to the older versions. By correlating the artifacts of newer version with the older ones, it shows how the application have been upgraded behind the scenes and by incorporating those results can provide investigators better understanding and insight for the certain evidence in a potential cybercrime case.*

**Keywords:** Telegram, Android, Digital forensics.

## INTRODUCTION

Various social media applications over the internet nowadays provide a variety of interesting set of features from texting, group chatting, notifications, location, contacts, file sharing to statuses updates all free of cost [1]. This is especially accelerated as with the rise of 4, 5G era, making the old Short Message Service (SMS) out of age [2].

According to [3] the internet has 3.97 billion user – almost half the world's population with the highest usage in Asia. Applications are readily available at the any user's ease offering services at various levels and industries. This increasing usage with the rise of digital services has led to the unlawful acts as well [4]. Any violation of law committed using computer technology is now being identifies as cybercrime [5][6][7]. One of the most serious of those crimes may include fraud, impersonation, and blackmailing.

In recent literature, smartphone forensics has been widely studied mainly focusing on Android and iOS platforms due to their pervasiveness [8]. As a result, a vast range of techniques and, methodologies are available that are used to extract and analysis the evidence from the smartphones [9]. We try to leverage these body of work for extraction and analysis of the Telegram application.

Anglano et al. 2017 [10] conducted forensic analysis on the Telegram application on Android. The authors reconstructed the contact list and messages that were exchanged between users. Using the logs file, they were able to map out the messages in chronological order with the details of who, when, the messages were exchanged and when they deleted. However, the hash functions were not used in this work.

Mahajan [11] carried out the forensic analysis on WhatsApp and Viber – two of the most popular instant messaging applications on the platform. They tried to find any data on the internal memory of the device i.e., messages or media files etc. However, the paper does not focus on the details of the artifacts or the evidence of their location.

*Received January 2, 2022; Revised January 25, 2022; Accepted February 3 ,2022*

Agrawal et al. 2019 [12] performed the forensic analysis on Facebook application on the virtualized android environment. They acquired the artifacts and performed user action and monitored the changes in device and over the network as well. The analysis only focused on the limited number of user activities.

Arshad et al. [13] explored the artifacts from Tor browser on the latest versions of Windows 10 and Android OS. They utilized storage, ADB logs. RAM and ZRAM on Android devices; a first-time analysis of an Android swap file. The investigation showed that a significant evidence extraction from these techniques can be achieved - approximately 60% and can be a considerable alternative to performing RAM investigations especially when there are privacy applications in question.

Asmara et al. [14] proposed a network forensic strategy for social messenger app named Signal that identify the artifacts from the encrypted network traffic. Proposed strategy can easily detect activities such as chatting, media messages, audio, and video calls by looking at the payload patterns from the from encrypted traffic.

Anwar et al. [15] investigated the location data collected by Google when the GPS is disabled on Android devices. They employed the live forensics on RAM with no evidence of such data, however they found some similarities in the data collected by the associated Google account which collects GPS locations from cellular networks, sensors and Wi-Fis with varying accuracy.

The paper focuses on the investigation into the digital forensics of social messaging applications available on Android OS smartphones [16]. These applications are available through Google Play Store and most of them are accessible free of cost. The goal of this paper to better help understand to forensic analysis of the application Telegram and the process through which and investigator may face during a cybercrime case.

The methodology of this paper can be summarized as follows:
1) Forensic methodology for an app running on Android smartphones.
2) Interpret the acquired artifacts from the analysis with predesigned user activities.
3) Furthermore, compare the results with the older version of the Telegram application.

In section II we map out our methodology used in the process. In, section III, we discuss the results of the forensic and finally, we present the conclusions of this work in section IV.

## METHODS

Given the objective of the forensic analysis the analysist must be allowed to obtain the evidence by the app. The analysis methodology we proposed can be categorized in four different sub parts.

### A. Identification

We consider the role of the writers as a forensic investigators or analyst [17]. This part of the methodology deals with the evidence identification. Where the said evidence is located or stored, what materials re being used as digital evidence what activities are being performed by the user on the application in question.

### B. Preservation

In this part we preserve the evidence acquired during the entire acquisition in the forensic analysis process. It is the most important part of the investigation where every artifact must be preserved the way it was originally found as any modification intentionally or accidently may cause inaccuracies in the analysis process further down the case and can even lead to the cancellation of a well-established digital forensic case.

The process for acquisition in this paper was performed by Android Debug Bridge (ADB) v31.0.2, Virtual Box v6.1, Gennymotion v.3.2.1, SQLite Database Browser v3.11.2 (Portable

version), Amaze file explorer and Notepad++ v7.5.4. The acquisition can only be done on a rooted Android smartphone.

### C. Analysis

After the digital evidence produced in the previous process of preservation, it must be read for the analysis. The data cannot, however, be read directly. The help of tools as mentioned previously were required and with those we were able to carry out the analysis process. The method we used for this was to compare the results of the previous analysis [10] that uses an older version of the telegram with the newest version that is now available today.

Fig.1 shoes the activities which are to be performed:

1) The initialization of the app: installation, signup, verification, launch.
2) CRUD operations on the contact: add, update, delete.
3) Messaging: sending text, media, contact, file, location sharing.
4) Furthermore, the uninstallation of the app.
5) Finally, the source code analysis and comparing all the above-mentioned parts one to one with the old version of the app.

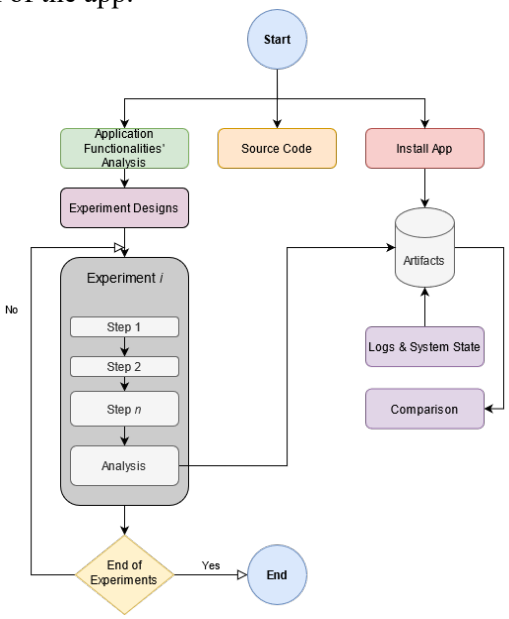


**Fig 1.** Workflow of the application analysis and individual experiments

### D. Presentation and Documentation

When all the previous mentioned steps have been carried out the final stage is to present the results and conclusions resulted in the process. The artifacts generated as a result would be compared with a previous version of the Telegram i.e., v3.15 which came out in 2016. The final object is to figure out what changes, if any, is there to the application and verify if there is any major change needed to carry out the forensic of the app now which will always help forensic analyst prove the cybercrime committed in a relevant court.

## RESULT AND DISCUSSIONS

### A. Setup and Preparations

In this paper, the scope was to determine the residual data from an Android device using sound forensic techniques on Telegram Messenger. User activities performed on the app is necessary to perform a thorough investigation such as: downloading and installation of the application, signing up into the app and then launching it for the first time; contacts creation, update, deletion; message exchanges including sending out the text message, receiving the

message, sending image; sharing contact and file. The test took place on a virtual Android smartphone configured with generic settings for eliminating the specificity of hardware.

For this work a virtual phone was used customized according to an average android smartphone configuration – 2GB of RAM, 2 Core processor, 16GB of storage with Android 8.1 having API level 27. Telegram Application version used was 7.7.2 (2293) which was the latest release of the forensic analysis. Fig 2. Shows the startup and signing in the app in virtual environment.

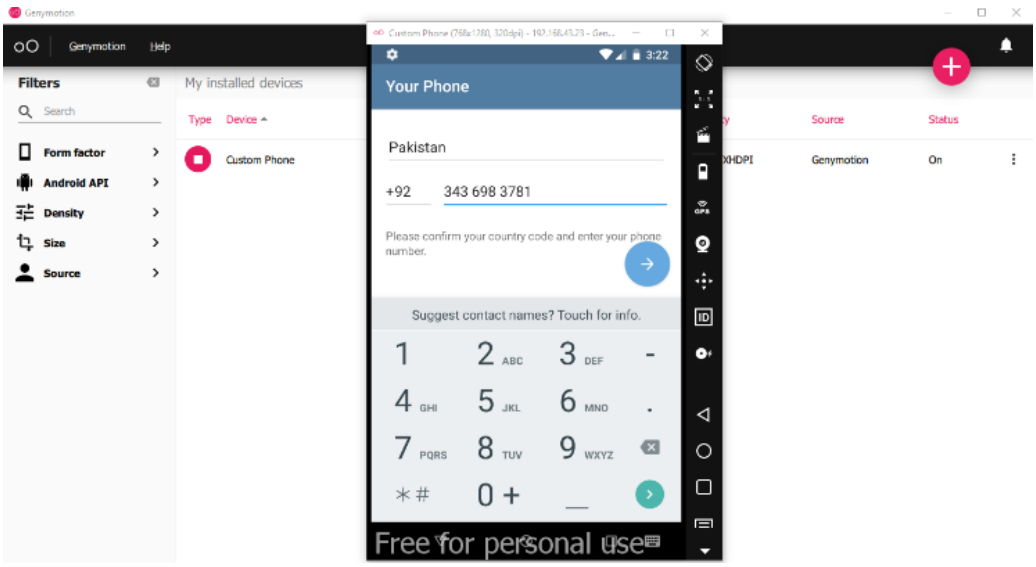


**Fig 2.** Virtual Smartphone Setup using Gennymotion v.3.2.1

For the acquisition of the artifacts, we used live and as well as online tools while the activities were happening in real time. We incorporated different tools to aid us to the access to a rooted virtual device, file explorer and command line interface for analyzing the logs of the events performed. At the end all the data generated along the activities were compiled to presented including the files, files' contents, and their locations.

## B. Investigation

### 1) File Structure

During the extraction of the data, we found different files and folders relating to the app directory in multiple location. Fig 3 shows the results of the findings. Firstly, *data/data/org.telegram.messenger.web* contains multiple files important to the evidence recovery. It stores all the files in the smartphone's local storage relating to the activities performed. The files folder contains the cache4.db file containing all the local cached database including tables having data of contacts, users, groups, messages etc. In contrast, *data/org.telegram.messenger* shows the normal directory in a non-rooted smartphone where the most important files and *shared_prefs* are not accessible.

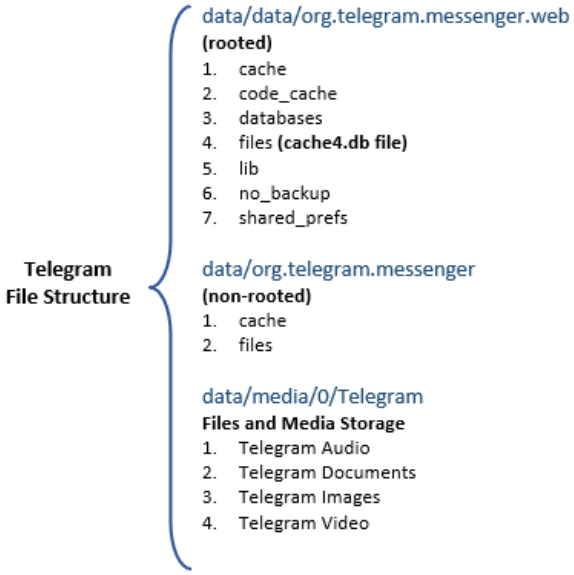


**Fig 3.** Telegram Files' Location and Structure

Finally, we have the location data/media/0/Telegram where the app stores its files and media. These are the files which are downloaded from the message exchanges and are stored on either the local or an external storage if present and selected as default; in case of external storage the path is <sd_card_name>/Telegram where sd_card_name is the name of the external card (without the angled braces) dependent on the model and make as shown in Fig 4. In comparison to the unrooted device, we found that these stored artifacts are sometimes stored on the cache folder temporarily.

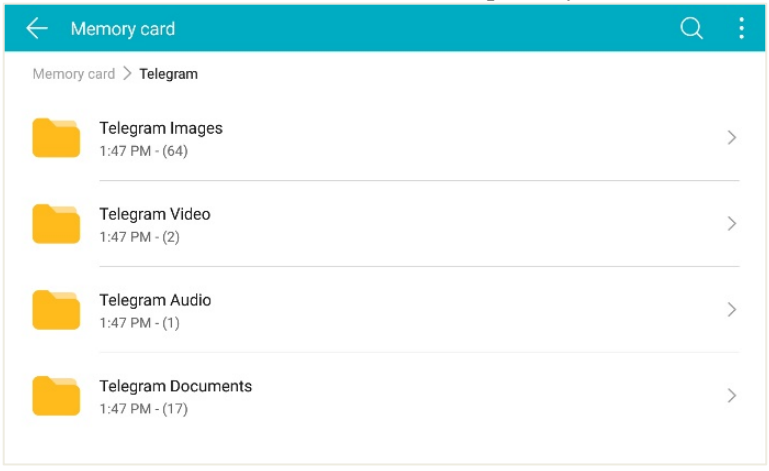


**Fig 4.** Files stored by Telegram in external memory

2) Initialization

The initialization included the downloading and installation of the application. After the installation, the app was launched for the time for the signup. The signup included OTP verification. We used ADB logcat for the online forensics and Genymotion own virtual device log generation for offline forensics.

The Fig. 5 shows the entry in the users table in the cache4.db file with the user who just signed into the application. This provide the evidence that the activity performed can

be proved through this method. We signed up with a mobile number which can clearly be (unencrypted) in blob seen in the right section of the screen capture. The unique id can also be seen for the user which is vital for the further investigation.

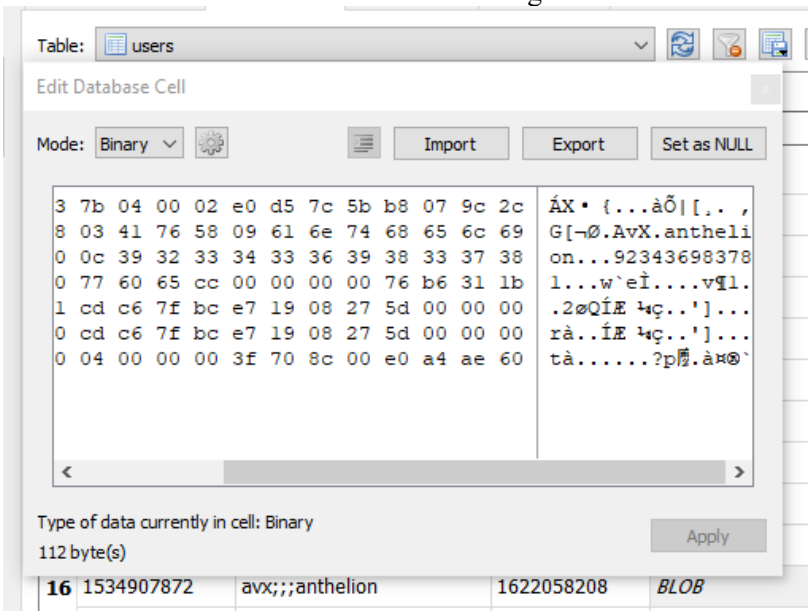


**Fig 5.** Database table showing user has signed into the app

The logs were obtained through *adb* logcat command with further filtration of the type of log we wanted in the analysis for example *adb -e logcat org.telegram.messenger.org* for filtering by the desired application processes and *adb -e *:D* for displaying all the debug process in the virtual mobile. All the logs for these activities are in Table 1. These logs can be useful as digital evidence in court later.

**Table 1.** App Setup and Initiation Logs

| Activity | Logs |
|---|---|
| Installation | 06-18 18:25:18.230 1286 1286 I Finsky : [5] VerifyInstallTask.mI(6): Verification complete: id=0, package_name=org.telegram.messenger.web |
| First Launch | 06-18 18:28:16.751 534 3187 I ActivityManager: START u0 {act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] flg=0x10000000 pkg=org.telegram.messenger.web cmp=org.telegram.messenger.web/org.telegram.ui.LaunchActivity} from uid 10019 |
| Verification | 06-18 18:28:17.787 534 3153 D VoldConnector: SND -> {8 volume mkdirs /storage/emulated/0/Android/data/org.telegram.messenger.web/cache/} |
| Opening | 06-18 18:30:35.214 534 3070 I ActivityManager: START u0 {cmp=org.telegram.messenger.web/org.telegram.ui.LaunchActivity (has extras)} from uid 10081 |

Regarding the comparison with the older version of the app, the file location and structure of the telegram files and folder including their names remain the same.

3) Contacts

As with the previous user activity, same method was utilized to obtain this data. Online log caught the relevant data during the performance of the activities related to the contacts where user first added the contact and then edited it and deleting it afterwards. It

could be used to strengthen the case in the case where a suspect might perform these activities.

The same location where the signed in user is stored in cache4.db is also where all the users and contacts can be seen. Each user has an id followed by a name. In Telegram a name is a display name (or nickname) visible to the header of a message thread in the app and multiple users can have the same display name. Whereas a username is a unique handle distinguishing each user from each other. Both display name and unique handle are stored in the users table separated by three semicolons as show in Fig 6. This is a crucial piece of artifact that shows which user holds which name and their corresponding handle which clearly distinguishes them for the other.

Table: users

| | uid | name | status | data |
|---|---|---|---|---|
| | Filter | Filter | Filter | Filter |
| 1 | 777000 | telegram;;; | 0 | BLOB |
| 2 | 136817688 | channel;;;channel_bot | 0 | BLOB |
| 3 | 586954814 | ;;; | -100 | BLOB |
| 4 | 698275978 | daywalker;;;bilalhassan1 | -100 | BLOB |
| 5 | 735150795 | weathery;;;weather... | 0 | BLOB |
| 6 | 962973735 | jin <grandfather>;;;ma... | -100 | BLOB |
| 7 | 1031714261 | muhammad raza shahid ... | 0 | BLOB |
| 8 | 1045761172 | bahram;;; | -100 | BLOB |
| 9 | 1188353915 | anonymous carlo;;;anon... | -101 | BLOB |
| 10 | 1242547689 | monochrome;;; | -100 | BLOB |
| 11 | 1271266957 | replies;;;replies | 0 | BLOB |
| 12 | 1316272383 | ijeoma okoli;;; | -100 | BLOB |
| 13 | 1351960791 | muhammad umer;;;pizz... | -100 | BLOB |
| 14 | 1431643403 | obama .;;;blackratguy | -101 | BLOB |
| 15 | 1469738945 | the ugly;;; | -100 | BLOB |
| 16 | 1534907872 | avx;;;anthelion | 1622058208 | BLOB |

**Fig 6.** Database table showing all the users as evidence

In case of deletion of a contact, user_contact_v7 with the fields userID, forename, surname is utilized with correlation to the table user_phones_v7 with the fields userID, phone, sphone, and deleted which shows which contacts once were saved were removed. Table 2. shows the log acquired through online forensic during the above-mentioned user activities.

**Table 2.** Contact Activity Logs

| Activity | Logs |
|---|---|
| Adding Contact | 06-18 18:38:42.300 1187 2697 E ContactsDatabaseHelper: Mimetypevnd.android.cursor.item/vnd.org.telegram.messenger.android.profile not found in the MIMETYPES table |
| Editing | 06-18 18:39:35.644 3783 3792 I chatty : uid=10081(org.telegram.messenger.web) FinalizerDaemon identical 1 line |
| Deleting Contact | 06-18 18:59:31.215 1069 1876 I NetworkScheduler.Stats: Taskcom.google.android.gms/com.google.android.gms.icing.proxy.IcingInternalCorporaUpdateService finished executing. cause:9 result: 1 elapsed_millis: 95 uptime_millis: 95 exec_start_elapsed_seconds: 2822 [CONTEXT service_id=218 ]file "/data/app/org.telegram.messenger.web-Wx9XXQMPqMBLM9OV7iw0Dw==/base.apk"],nativeLibraryDirectories=[/data/app/org.telegram.messenger.web-Wx9XXQMPqMBLM9OV7iw0Dw==/lib/x86, /data/app/org.telegram.messenger.web- |

| | |
|---|---|
| | Wx9XXQMPqMBLM9OV7iw0Dw==/base.apk!/lib/x86, /system/lib, /system/vendor/lib]] |
| Opening | 06-18 18:30:35.214 534 3070 I ActivityManager: START u0 {cmp=org.telegram.messenger.web/org.telegram.ui.LaunchActivity (has extras)} from uid 10081 |
| Uninstall | 06-18 18:23:54.900 1286 1286 I Finsky : [5] sjg.s(3): Package no longer installed: org.telegram.messenger.web |

Regarding the comparison with the older version of minor changes are there within the cache4.db database file i.e., the table names user_phone_v3 and user_contact_v3 are now changed to user_phones_v7 and user_contact_v7 respectively which seems to change along with app's version number.

4) Chat Messages

In this scenario, there were a few use cases performed – first, a simple text message was sent to one of the contacts; secondly, an image was shared; after that, a media file was sent followed by sharing a saved contact; finally, location was shared. All the logs were accessed the same way as with the previous cases – through ADB and Genymotion's own log generation. Thought all the logs were massive in number filtering them out based on the timestamp were necessary.

Fig 7. shows the messages table in the cache4.db file. Entry number 17 shows the uid with 1188353915 identifying the user who sent the message and the blob displaying the message itself unencrypted. This simple text is useful to the investigator finding out the actual content of the message and other related information such as timestamp and whether the message was sent or read until the acquisition.

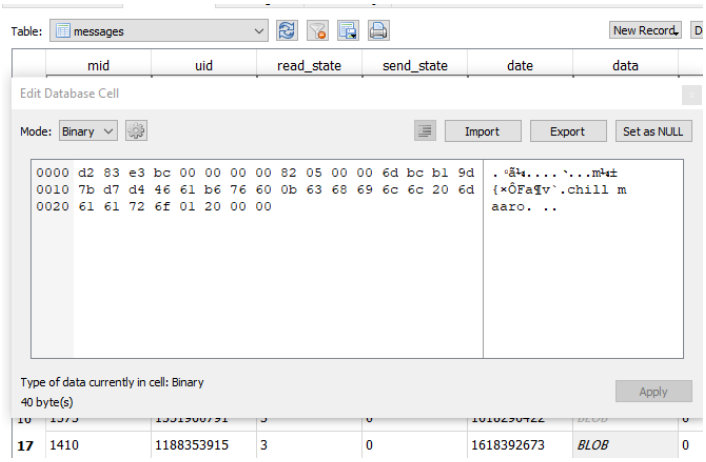


**Fig 7.** Screenshot showing entry in the cach4.db for a text message

Although any plain text message is visible in the SQLite database browser, the media message is however, not readable. As described in [18] the message's media file name is visible in Telegram v3.4.2 it is not, however, the case with our analysis as shown in Fig. 8. It is to be noted that any caption relating to a media message can be seen in the section above and in some cases the snippet including in a URL is also saved in the database in media_v2 table.

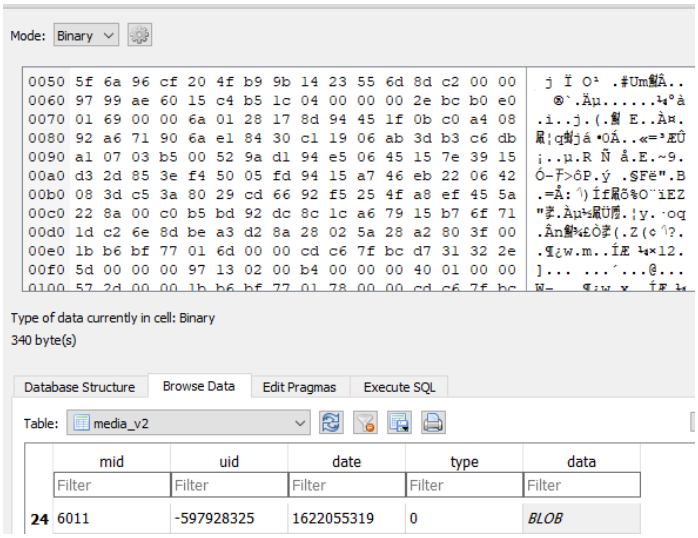


**Fig 8.** Database showing contents of a media message

The logs were captured using the online forensics in the message exchanges as can be seen in Table 3.

**Table 3.** Evidence of Logs in Message Exchanges

| Activity | Logs |
|---|---|
| Sending Plain Text | 06-18 19:05:09.601 3783 3788 I zygote : Compiler allocated 14MB to compile void org.telegram.messenger.SendMessagesHelper.sendMessage |
| Receiving Plain Text | 06-18 19:06:52.528 391 4954 D AudioTrack: Client defaulted notificationFrames to 8169 for frameCount 24508 |
| Sending Image | 06-18 19:06:14.152 3783 3788 I zygote : Compiler allocated 9MB to compile void org.telegram.ui.Cells.DialogCell.buildLayout() |
| Sending File | 06-18 19:08:55.553 3783 3783 I zygote : Deoptimizing void org.telegram.messenger.ImageReceiver.setImage(org.telegram.messenger.ImageLocation, java.lang.String, org.telegram.messenger.ImageLocation, java.lang.String, org.telegram.messenger.ImageLocation, java.lang.String, android.graphics.drawable.Drawable, int, java.lang.String, java.lang.Object, int) due to JIT same target |
| Sending Contact | 06-18 19:10:22.254 471 471 D SurfaceFlinger: duplicate layer name: changing org.telegram.messenger.web/org.telegram.ui.LaunchActivity to org.telegram.messenger.web/org.telegram.ui.LaunchActivity#2 |
| Deleting Message | 06-18 19:07:49.550 3783 3783 I zygote : Deoptimizing void org.telegram.ui.ActionBar.ActionBarMenu.hideAllPopupMenus() due to JIT inline cache |

5) Source Code

Telegram uses TDS (Telegram Data Structure) to encode the information it stores. Most of the user's information the application stores are in the shared preferences which is the mechanism to save states to the application after it has been terminated in Android OS e.g., appLocked, loginTime, lastMyLocationShareTime, sharingMyLocationUntil, passcodeRetryInMs etc. As discussed in the first section of the investigation process in File Structure section the shared preference is stored in the shared_prefs>userconfig.xml. One of the most important pieces of information the stores is the user information in a serialized form.

As shown in Fig. 9 one of the data types is the string with the user value for the user to which the account belongs to. Telegram allows the feature to add more than one account in which separate user configuration file is generated with names userconfig1 and userconfig2 and so on. These serialized strings are then deserialized using the TDS extracted from the source code. The deserialization was not performed in our process as this was not part of our proposed methodology. However, [19] provides all the schema related to the data structures and all the other variables the application uses inside the code. The source code provided [20] us useful in corelating the java classes with TDS which are then can be used for the deserialization process.

```
<string
name="user">wViEk3sEAALg1XxbuAecLEdbrNgDQXZYCWFud
GhlbGlvbgAADDkyMzQzNjk4Mzc4MQAAAHdgZcwA&#10;AAAAd
rYxGxgy+FHNxn+85xkIJ10AAABy4AUAzcZ/vOcZCCddAAAAdO
AFAAQAAAA/cIwA4KSuYA==&#10;    </string>
<int name="2dialogsLoadOffsetDate" value="0" />
```

**Fig 9.** Serialized String in userconfig.xml file

Regarding the comparison of the old and newer versions of the application we found some changes being made to the TDS e.g., the case with user's details the parameters used are id, first name, last name, phone, photo, status, username as shown in Fig. 10 used to deserialize the string is unchanged according to [https://core.telegram.org/schema]. As with the case of java classes the source only provides the code up until v5 and cannot therefore compare it accurately. However, comparing the analyzed code in [10] with the v3, the class names does seem to be retaining their names.

**Parameters**

| Name | Type | Description |
|---|---|---|
| id | int | User identifier |
| first_name | string | First name (see below) |
| last_name | string | Last name (see below) |
| access_hash | long | Checksum, dependant on user ID |
| phone | string | Phone number |
| photo | UserProfilePhoto | Profile photo |
| status | UserStatus | Current status |
| username | string | Username<br>Parameter added in Layer 18. |

***Fig 10.*** *Name, their Data Type and Description for a User TDS*

The analysis provided is guide that can be utilized by the investigators and forensic analyst that could improve knowledge for cyberlaw professionals in finding artifacts on Telegram Messenger application. We obtained some important data by performing some pre-designed user activities on the app that gave us insight to how the application behaves when certain actions are performed.

Table 4 shows the precise comparisons between the work conducted on the Telegram v3.15 in 2017 and our four years later with the v7.7.2. With change to the application version the Android OS itself has upgraded so it was important that we

conducted our investigation on a newer version of the smartphone; in comparison we used Android v8.1 to the older version v.4.4 and v7.1. Similarly, instead of using an actual physical smartphone we setup a virtual environment to try and dig out the artifacts and weigh out how much they differ. Finally, with the acquisition tools we used ADB in comparison to the propriety Cellebrite UFED4PC in the original work.

**Table 4.** Comparison in the Forensic Setup Performed in [10] and in this work

| Telegram v3.15 [10] | Telegram v7.7.2 |
|---|---|
| Android Mobile Device Emulator | VirtualBox and Genymotion |
| Android v4.4-7.1 | Android v8.1 |
| Samsung Galaxy Core Plus | Custom Phone |
| Cellebrite UFED4PC | Android Debug Bridge |

The gathered data was then put in comparison with the results of the forensics of the old version of application. We compared the app artifacts' locations, performed actions related to contact's addition, modification and deletion and difference type of messages were exchanged. Upon the analysis we found that some of the database's table names were different such as user_phone_v3 and user_contact_v3 are now changed to user_phones_v7 and user_contact_v7 respectively which seems to change along with app's version number.

Furthermore, the case with the message exchanges involving media, the location to where the media was stored on the phone was unreadable. In the case of the source code analysis, we were able to compare the v3 to v5 code and did not find any major changes such as different java classes' name and their location in the java package.

We present a brief guide to our findings at the end of our investigation. Table 5 shows two columns – the activities we performed and the locations that had an effect from those activities. In short, either the database file was updated in case of activities with signing in, contacts and plain messages. Whereas the location where all the media and files are stored was impacted with the activities involving message exchanges with images, location or contact sharing. This guide can provide insight to the investigators when looking for the certain evidence in a potential cybercrime case.

**Table 5.** Activities Performed and Their Impacted Location on Telegram

| Activity | Locations |
|---|---|
| Signing up | \data\data\org.telegram.messenger\files\cache4.db |
| Uninstallation | \data\media\0\Telegram* |
| Contact Add | \data\data\org.telegram.messenger\files\cache4.db |
| Contact Modify | \data\data\org.telegram.messenger\files\cache4.db |
| Contact Deletion | \data\data\org.telegram.messenger\files\cache4.db |
| Text Message | \data\data\org.telegram.messenger\files\cache4.db |
| Media Message | \data\media\0\Telegram* |
| Location Sharing | \data\media\0\Telegram* |
| Contact Sharing | \data\media\0\Telegram* |

## CONCLUSIONS

Based on the results obtained by the five major scenarios we conducted on Telegram Messenger v7.7.2 on Android OS 8.1, this paper lays out the details of the artifacts and addition to those it

compares them to the older version as well. This not only provide a clearer perspective to the forensic analysis to the application but also tries to map out the functionalities of the app, how they impact the data and updates the forensic process accordance to the newer version. As a conclusion, this work can be used by the digital forensic analyst on the field as a reference relating to the data discovery and artifacts in a crime scene.

## REFERENCES

[1] M. Landwehr, A. Borning, and V. Wulf, 'The High Cost of Free Services: Problems with Surveillance Capitalism and Possible Alternatives for IT Infrastructure', in *Proceedings of the Fifth Workshop on Computing within Limits*, Lappeenranta Finland, Jun. 2019, pp. 1–10. doi: 10.1145/3338103.3338106.

[2] C. Q. Lau, A. Cronberg, L. Marks, and A. Amaya, 'In Search of the Optimal Mode for Mobile Phone Surveys in Developing Countries. A Comparison of IVR, SMS, and CATI in Nigeria', *Survey Research Methods*, pp. 305-318 Pages, Dec. 2019, doi: 10.18148/SRM/2019.V13I3.7375.

[3] T. text provides general information S. assumes no liability for the information given being complete or correct D. to varying update cycles and S. C. D. M. up-to-D. D. T. R. in the Text, 'Topic: Internet usage worldwide', *Statista*. https://www.statista.com/topics/1145/internet-usage-worldwide/ (accessed Mar. 07, 2022).

[4] C. Ruiz-Mafe, E. Bigné-Alcañiz, and R. Currás-Pérez, 'The effect of emotions, eWOM quality and online review sequence on consumer intention to follow advice obtained from digital services', *JOSM*, vol. 31, no. 3, pp. 465–487, Jun. 2020, doi: 10.1108/JOSM-11-2018-0349.

[5] Department of IT Convergence Engineering, School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, South Korea, G. B. Satrya, and S. Y. Shin, 'Optimizing Rule on Open Source Firewall Using Content and PCRE Combination', *JACN*, vol. 3, no. 4, pp. 308–314, 2015, doi: 10.18178/JACN.2015.3.4.188.

[6] G. B. Satrya, N. D. W. Cahyani, and R. 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*, Seoul, Republic of Korea, 2015, pp. 1–4. doi: 10.1145/2781562.2781567.

[7] 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)*, Vienna, Austria, Jul. 2016, pp. 430–435. doi: 10.1109/ICUFN.2016.7537064.

[8] S. C. Sathe and N. M. Dongre, 'Data acquisition techniques in mobile forensics', in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, Coimbatore, Jan. 2018, pp. 280–286. doi: 10.1109/ICISC.2018.8399079.

[9] L. D. Turner *et al.*, 'Evidence to support common application switching behaviour on smartphones', *R. Soc. open sci.*, vol. 6, no. 3, p. 190018, Mar. 2019, doi: 10.1098/rsos.190018.

[10] C. Anglano, M. Canonico, and M. Guazzone, 'Forensic analysis of Telegram Messenger on Android smartphones', *Digital Investigation*, vol. 23, pp. 31–49, Dec. 2017, doi: 10.1016/j.diin.2017.09.002.

[11] A. Mahajan, M. S. Dahiya, and H. P. Sanghvi, 'Forensic Analysis of Instant Messenger Applications on Android Devices', *IJCA*, vol. 68, no. 8, pp. 38–44, Apr. 2013, doi: 10.5120/11602-6965.

[12] A. K. Agrawal, A. Sharma, and P. Khatri, 'Digital Forensic Analysis of Facebook App in Virtual Environment', *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, Mar. 2019.

[13] M. R. Arshad, M. Hussain, H. Tahir, S. Qadir, F. I. Ahmed Memon, and Y. Javed, 'Forensic Analysis of Tor Browser on Windows 10 and Android 10 Operating Systems', *IEEE Access*, vol. 9, pp. 141273–141294, 2021, doi: 10.1109/ACCESS.2021.3119724.

[14] A. Afzal, M. Hussain, S. Saleem, M. K. Shahzad, A. T. S. Ho, and K.-H. Jung, 'Encrypted Network Traffic Analysis of Secure Instant Messaging Application: A Case Study of Signal

Messenger App', *Applied Sciences*, vol. 11, no. 17, p. 7789, Aug. 2021, doi: 10.3390/app11177789.
[15]N. Anwar, M. M. Mardhia, and L. Ryanto, 'Live Forensics on GPS inactive Smartphone', *mob.forensics.j*, vol. 3, no. 1, pp. 32–44, Mar. 2021, doi: 10.12928/mf.v3i1.3847.
[16]N. Al Mutawa, J. Bryce, V. N. L. Franqueira, A. Marrington, and J. C. Read, 'Behavioural Digital Forensics Model: Embedding Behavioural Evidence Analysis into the Investigation of Digital Crimes', *Digital Investigation*, vol. 28, pp. 70–82, Mar. 2019, doi: 10.1016/j.diin.2018.12.003.
[17]B. O. Gardner, S. Kelley, D. C. Murrie, and I. E. Dror, 'What do forensic analysts consider relevant to their decision making?', *Science & Justice*, vol. 59, no. 5, pp. 516–523, Sep. 2019, doi: 10.1016/j.scijus.2019.04.005.
[18]T. Hermawan, Y. Suryanto, F. Alief, and L. Roselina, 'Android Forensic Tools Analysis for Unsend Chat on Social Media', in *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia, Dec. 2020, pp. 233–238. doi: 10.1109/ISRITI51436.2020.9315364.
[19]'Schema'. https://core.telegram.org/schema (accessed Mar. 07, 2022).
[20]DrKLO, *DrKLO/Telegram*. 2022. Accessed: Jan. 07, 2022. [Online]. Available: https://github.com/DrKLO/Telegram