Analyzing TikTok from a Digital Forensics Perspective*

Patricio Domingues[1,2†], Ruben Nogueira[1], José Carlos Francisco[1], and Miguel Frade[1,3]
[1]ESTG/Polytechnic of Leiria, Leiria, Portugal
patricio.domingues@ipleiria.pt, {2171569, 2202274}@my.ipleiria.pt, miguel.frade@ipleiria.pt
[2]Instituto de Telecomunicações, Leiria, Portugal
[3]CIIC Research Centre, Leiria, Portugal

Received: January 4, 2021; Accepted: May 3, 2021; Published: September 30, 2021

Abstract

TikTok is a major hit in the digital mobile world, quickly reaching the top 10 installed applications for the two main mobile OS, iOS and Android. This paper studies Android's TikTok application from a digital forensic perspective, analyzing the digital forensic artifacts that can be retrieved on a post mortem analysis and their associations with operations performed by the user. The paper also presents FAMA (Forensic Analysis for Mobile Apps), an extensible framework for the forensic software Autopsy, and FAMA's TikTok module that collects, analyzes, and reports on the main digital forensic artifacts of TikTok's Android application. The most relevant digital artifacts of TikTok include messages exchanged between TikTok so-called "friends", parts of the email/phone number of registered users, data about devices, and transactions with TikTok's virtual currency. One of the results of this research is the set of forensic traces left by users' transactions with TikTok's in-app virtual currency. Another result is the detection of patterns that exist in TikTok's integer IDs, allowing to quickly link any 64-bit TikTok's integer ID to the type of resources – user, device, video, etc. – that it represents.

Keywords: TikTok, Android, Digital Forensics, In-app Virtual Currency

1 Introduction

TikTok is a social media platform which has taken the mobile world by storm. With less than four years of existence in the international market, TikTok reached the top 10 of most downloaded applications for mobile devices in the 2010-2019 decade [28], even achieving rank #2 in 2019 [19], and credited with around 800 million active users [31].

The concept of TikTok is simple and centered around combining an endless stream of short duration videos, ranging from 15-second to 60-second long, of all kinds, often with music. On top of video sharing, TikTok has social network features, such as the concept of followers/friends, the possibility to comment on videos, exchange messages, perform live transmissions, and to donate gifts bought with TikTok virtual currency.

The origins of TikTok stem from two other services: `Musical.ly` and `Douyin`. The former – `Musical.ly` – was a popular mobile application released in 2014 centered around 15-second lip syncing videos, making available to video creators a wide library of high quality commercial music [24]. In the third

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 12(3):87-115, Sept. 2021
DOI:*10.22667/JOWUA.2021.09.30.087*

*Extended version of "Post-mortem digital forensic artifacts of TikTok Android App" [12]

†Corresponding author: ESTG / Polytechnic of Leiria, Tel: 2411-901 Leiria, Portugal Email: patricio.domingues@ipleiria.pt

quarter of 2016, a similar sync-lip service appeared in China, under the name of Douyin, with ByteDance, the company behind Douyin, establishing TikTok as the international (*i. e.*, non-Chinese) version of Douyin [1]. In late 2017, ByteDance acquired and merged `musical.ly` with TikTok. Nowadays, both `TikTok` and `Douyin` continue to exist in their respective markets [16], while `musical.ly` has been totally integrated into TikTok. Interestingly, the URL `musical.ly` redirects to `TikTok.com`. Lately, TikTok has been subjected to some instability due to non-technical reasons [2, 3], and has attracted its fair share of controversy [4, 1].

TikTok appeals and targets a young population, mostly the pre-adolescent (10-14 years old) and adolescent (15-19 years old) groups. This is attributed not only to the short duration of videos, but also to the empowerment brought by TikTok to young individuals, as it eases the creation of videos with background audio and visual effects, all through the smartphone, which is a central tool in the life for the so called digital natives [8]. Moreover, there is also the attractiveness that a video can reach viral status, bringing recognition and revenues to creators. TikTok seems particularly appealing to the digital born generation [8].

The importance of digital forensic for TikTok is manyfold. First, the young demographic of TikTok can attract crime against children, such as online enticement, sexual exploration and sex extortion [7]. Secondly, due to the easiness of creating and publishing videos, TikTok is sometimes used by extremist groups to publish hate material [35]. The absence of a real search feature, as searches for TikTok videos need to be made by selecting hashtags, hardens the detection of inappropriate content by law enforcement agents. Thirdly, analyzing the interactions of an individual with TikTok – published videos, liked videos, friends in the network, exchanged messages, donated gifts – might provide valuable insight of user's tastes and way of thinking, besides providing evidence of his/her actions.

All of these contribute to the motivation for this work, whose main goals are to *i*) identify the main forensic artifacts of TikTok application for Android and to *ii*) provide an open-source software tool to extract, analyze and report on these forensic artifacts.

This paper extends our previous work [12], which studied TikTok's digital forensic artifacts available in version 16.0.41 (May 2020), whereas this work now focuses on version 18.1.3 (December 2020). This extended version adds the following main contributions: 1) Revision and validation of the digital forensic artifacts of TikTok's in the newer studied version; 2) Analysis of TikTok's 64-bit integer ID scheme, decoding the different types of identifiers, allowing for a quick detection of resource types (accounts, videos, devices, *etc*) and the associated timestamps; 3) Study of the forensic artifacts related to the usage of TikTok own coins and virtual gifts, namely purchasing coins, donating and receiving virtual gifts; 4) Presentation of our Forensic Analysis for Mobile Apps (`FAMA`), a new open-source framework to collect, process and report Android's digital forensic artifacts of Android smartphones; 5) Analysis of the TikTok module for FAMA, namely its main outputs when dealing with TikTok artifacts.

The remainder of this paper is organized as follows. Section 2 reviews related work, while Section 3 presents the TikTok ecosystem. Section 4 analyzes the multiple forensic artifacts provided by the Android TikTok app, while Section 5 studies artifacts linked to TikTok coins and virtual gifts usage. In Section 6, the forensic framework for the Autopsy software is presented, with emphasis to the application of the framework for TikTok Android app. Finally, Section 7 concludes the paper and shows some possible venues for future work.

2 Related Work

Khoa et al. analyze TikTok for Android from a digital forensic perspective [14]. They describe the main artifacts for version 8.9.4 of TikTok for Android. However, contrary to our work, they do not propose any tool to ease the task of digital forensic practitioners charged to analyze TikTok. Moreover, TikTok

has evolved substantially since version 8.9.4, as our work addresses version 18.1.3. In fact, as we report in this paper, we noted differences between our previous work, with version 16.0.41 [12] and the now studied version.

Benson observes that the posting timestamp of a TikTok video can be determined by its URL [5, 6]. Specifically, the 32 most significant bits of the 64-bit ID of a TikTok video corresponds to the date/time of when it was posted to TikTok. The timestamp is expressed in UNIX Epoch format. Additionally, the author's online service Unfurl[1] decomposes the data comprised within an URLs of TikTok's videos, namely the account of the video, as well as the post date/time. Note that TikTok currently displays, on the web interface, the posting date (not the hour) of a video, when the video is being shown, although the precise date/time can be found in the HTML of the page [5]. Benson also hints on the pattern of the lowest 32 bits of TikTok's ID of videos, hinting that they could be *flags and map to an internal lookup table*. In this work, we further analyze TikTok IDs, detecting patterns in the least significant 32 bits. These patterns allow us to quickly classify the type of resources – account, video, device, etc. – represented by a given ID.

Robert analyzes network traffic generated by TikTok, namely an HTTPS Post message that contains clear text plus an encrypted part [25]. This message is sent every five minutes by TikTok. The clear text contains a large set of parameters, regarding 1) the device (*e. g.*, screen resolution), 2) the user and 3) the application. In a followup post, the author analyzes the encrypted content by intercepting the call to the encryption method, observing that it is comprised of JSON-formatted data that include information regarding the device, and events logging [26]. Robert classifies as "not really personal" the encrypted data periodically sent by the application to TikTok servers. Likewise, Tidy [31] observes that the TikTok mobile application collects data about the watched videos, the location, phone model and operating system, as well as all touch activity within the application, that is, the keystroke dynamics. Our work confirms that these data and other events are collected by the mobile app, as it is stored in its databases and XML files, as we shall see later.

The anonymous author *BTF_117* provides a detailed Open Source Investigation (OSINT) of TikTok through several blog posts, with the main goal of studying the possibility of gathering data about a specific TikTok user [10]. In his detailed research, the author sets a so-called man-in-the-middle infrastructure to intercept the HTTPS network traffic exchanged between the client sides of TikTok – both the web interface and the phone app are studied – and TikTok servers. Captured traffic is mostly JSON-formatted data that provides a significant amount of information about the targeted user, including personal data and published video. Contrary to BTF_117's work, our study focuses on the content that exists in the phone. We believe that both approaches are complementary and can provide valuable data when combined.

In his digital forensic blog, Brignoni [9] analyzes the forensic artifacts of TikTok, focusing on message exchange and on some XML files. Our observations confirm that Brignoni's results regarding messages are still valid, indicating some stability of the app in the way it deals and stores messages, as Brignoni conducted the study in 2018.

Fergus et al. scrutinize TikTok and WeChat from a privacy point of view [27]. They observe that version 17 of TikTok's mobile application was collecting the local IP address of the device, along with the IP addresses of the DNS servers. The work also comments on the clipboard access for reading detected in the iOS version of the application, remarking that it was caused by an third-party SDK linked to ads, and that the same behavior was observed in many other mobile applications for the same reasons. Another issue with clipboard access was labelled as anti-spam feature, as TikTok sought to detect users that were repeatedly dumping the clipboard content as comments, either for self promotion or simply for nuisance [27]. The report also mentions that plaintiffs declared in a class action lawsuit that previous

[1] `https://dfir.blog/unfurl/`

versions of TikTok were collecting device identifiers such as the International Mobile Equipment Identity (IMEI) and the International Mobile Subscriber Identity (IMSI) numbers. However, the authors verified that both the mobile applications and the web platform were not collecting these data as of August 2020 [27]. As we shall see later, our work confirms that version 18.1.3 no longer stores the IMEI and the IMSI identifiers.

Neyaz et al. forensically review the Android versions of FaceApp and TikTok, analyzing privacy issues, needed OS permissions, and the network traffic [20]. The authors also used steganography in TikTok (version 12.3.5) to inject hidden text messages into videos posted to TikTok to check whether hidden text could be retrieved by TikTok viewers. They concluded that TikTok processing garbled the hidden content of videos, rendering useless the attempted steganography technique.

Kaye et al. compare Douyin and TikTok [16], focusing on infrastructures, features, business models, and governance. Wang [34] provides a study, mostly based on graphical representation, to highlight the data dimension and wide spread of TikTok.

Pandela and Riandi study the digital forensic artifacts retrieved when several operations are performed on a TikTok account accessed through a web browser [22]. They resort to three main tools: *FTK*, *Browser History Capture*, and *Video Cache View*. They conclude that login name, some text and photos thumbnails, but no videos, can be retrieved from the analysis of the browser. In this work, we focus solely on TikTok Android application.

3 TikTok Ecosystem

In this section, we review the main elements of what we call the TikTok ecosystem: *i*) TikTok's resource identifiers; *ii*) ways of accessing TikTok; and *iii*) the main features of TikTok.

3.1 Resource identifiers

TikTok resorts to 64-bit integers to identify several types of resources, namely accounts, videos, devices, live sessions, and installation. Each 64-bit integer encompasses in the most significant 32 bits, a universal time coordinated (UTC) timestamp expressed in Unix EPOCH format [5], and an ID in the least significant 32 bits. By analyzing the 32 least significant bits of TikTok identifiers, we detected a pattern that links an identifier to the type of resource it represents, allowing us to discern between an account ID, a video ID, a live ID, and so on. The pattern can be easily spotted when the 32 least significant bits of an ID are represented in hexadecimal format, as the third lowest hexadecimal symbol assumes a value according to the type of resource. For instance, IDs of videos have a `D` in the third-lowest position of their hexadecimal representation. An example is ID `6912215720931757317`, which is `0x5FED1F936102 0D05` in hexadecimal (see Figure 1). The highest 32 bits – `5FED1F93` –, that is, 1609375635 in decimal, encode the timestamp *Thu Dec 31 2020 00:47:15 GMT+0*, which corresponds to the date/time when the video was uploaded to TikTok. The `D`, which is the third least significant hexadecimal symbol in the lowest 32 bits of the identifier – `6102 0D05` – means that the resource is a video[2].

The presence of a timestamp in TikTok's IDs is of great relevance in digital forensics as it links a date/time to the resource that it identifies. This way, one can easily determine when a TikTok account was created, the date/time of a video upload to the social network, or when a live session was started. A minority of account IDs do not follow the 32-bit + 32-bit pattern, and thus the date/time cannot be inferred from the highest 32-bits. Based on the fact that some of these accounts have their first video

[2]This can be confirmed by accessing `https://www.tiktok.com/@nature2admire/video/6912215720931757317`

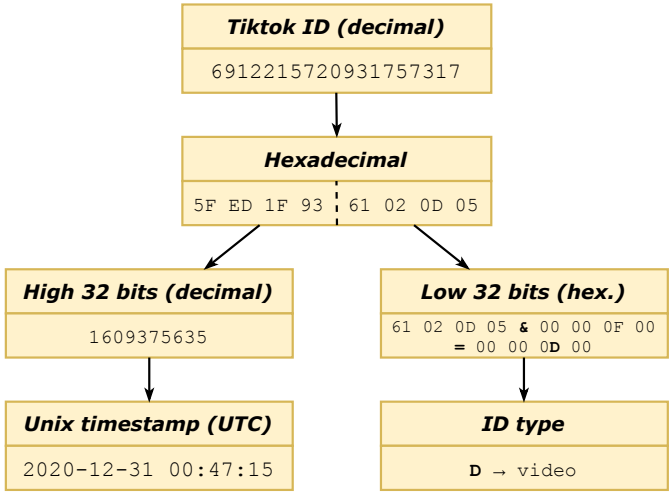


Figure 1: Decoding the timestamp and ID type from a TikTok ID value.

Table 1: Patterns for TikTok resources identifiers

Resource	Pattern (3^{rd} lowest hex)	Description
`device_id/wid`	6	devices (smartphone, web browser)
`room_id`	B	live sessions
`userID`	0 or 4	accounts
`video_id`	D	videos

published in 2017, we hypothesize that these accounts were created in `musical.ly` and then migrated to TikTok when `musical.ly` was merged with TikTok.

Table 1 lists TikTok's types of identifiers and the corresponding 3^{rd} least significant hexadecimal value for each pattern. The name for each type of identifier, given in the left column of the table, follows the designation found in the logs of TikTok's mobile application. Note that account IDs (`user_id` in Table 1) can have either a `0` or a `4` as pattern identifier. Furthermore, the designation `wid` references the ID assigned by TikTok to a web browser when one accesses TikTok's website. This ID can be seen in the page code received by the browser with the label `wid` and seems linked to the browser's sessions, as cleaning cookies originates the attribution of a new `wid` to the browser.

3.1.1 Account identifiers

Within TikTok, a registered user has three distinct identifiers: *i*) a `userID` also referred as `UID`, which is the 64-bit integer ID sets by TikTok when the account is created; *ii*) an alias name set by the registered user; and *iii*) a `uniqueID`, referred as the `username`, which can be used with the `@` symbol to reference the user within the TikTok network. The `uniqueID` is also set by the account owner, and can be changed, although as of December 2020, the minimum time interval between two successive changes is 30 days. Table 2 lists the identifiers for the `@nature2admire` TikTok account. Note that the `userID` `69121917 00166542342` encodes, as shown earlier, in 32-bit Unix EPOCH format, the timestamp of the account creation in the higher 32 bits, "`2020-12-30 23:14:02`" in this case. The hexadecimal representation of the lowest 32-bit of the identifier – 0x9BD48406 – also confirms our previous observation that account

Type	Example
userID (UID)	6912191700166542342
nickname	user981414799249
username (uniqueID)	nature2admire

Table 2: Identifiers for the @nature2admire TikTok account.

Listing 1: HTML code with TikTok ID of a given account

```
"author":{"id":"6912191700166542342",...,
"uniqueId":"nature2admire",
"nickname":"user981414799249",...}
```

IDs have either a 0 or a 4 in the third least significant symbol.

3.2 Accessing TikTok

TikTok is accessible via the web at tiktok.com, where anyone can watch videos or share them to other social networks such as Twitter, Facebook, *etc*. As soon as the browser opens TikTok.com, a video selected by the TikTok algorithm is played in a loop. Scrolling down the page brings other videos on an endless stream. For closer interactions, such as liking, commenting/reading the comments on a video, or referencing another account, one needs to be logged into a TikTok account.

To view, through the web, the public content of a given account , one needs to add the account's uniqueID to TikTok.com. For example, TikTok account nature2admire is accessible through the address https://tiktok.com/@nature2admire, where one can watch the public videos (if any) of the account. Alternatively, the same can be achieved with the userID, that is, https://tiktok.com/@6912191700166542342. Note that the TikTok identifiers – userID, nickname and uniqueID can be found in the HTML code of the web page of the account (see Listing 1).

Despite its support for the web, TikTok's main targets are smartphones. It has versions for the two major mobile OSes, Android and iOS. In this paper, we focus on the Android version of the application (henceforth *TikTok app* or *app*). While the app allows for video capture, video edition, video effects, and adding audio from a large library of commercial music, along other functionalities, TikTok web interface only provides for the uploading of videos, with no support whatsoever for edition/effects/audio and alike. The TikTok app requires an internet connection, as it only provides limited functionality when no internet connection is available.

To create an account within TikTok, one needs to provide either an email address or a valid phone number. Interestingly, the app supports multiple TikTok accounts. Switching between accounts within the app is also straightforward: the user only needs to select the account to switch to through the drop-down menu seen at the center top of the screen shown in the left side of Figure 2. The right side of Figure 2 shows the TikTok app displaying a video.

3.3 Main features of TikTok

The main features of TikTok are linked to its core elements: videos and music. Watching videos can be done in an ad-hoc manner, where TikTok's internal algorithm selects videos and streams them to the device until it is stopped by the user. Although TikTok provides a search engine for videos, the search is performed on the text that describes videos, in hashtags (*e. g.*, #nature), and in account names. Hashtags

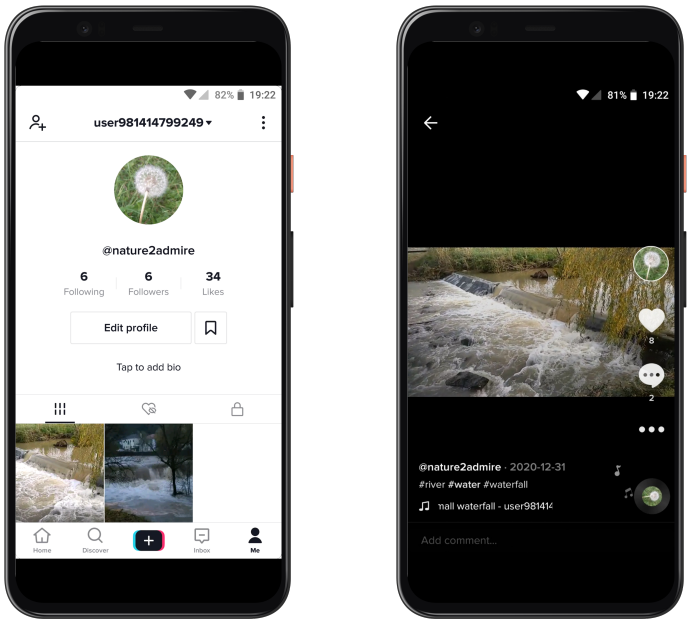


Figure 2: TikTok app displaying the main screen for the `@nature2admire` account (left), and displaying a video (right)

for a given video are defined by the owner of the video. In videos with music, while the video is being displayed on the device, a music identifier with a resemblance to a vinyl disk is shown rotating at the bottom right of the screen. By clicking on it, TikTok opens a view with a list of videos that use the same music. This is yet another way to access videos that share a common property.

Besides posted videos, accounts that have 1 000 or more followers can perform live shows, that is, streaming a real-time video feed. Followers can interact with the live feed, commenting, and sending gifts in the form of emojis and stickers. These virtual gifts need to be purchased with TikTok's own digital currency named *coins*, which users can buy from within the TikTok application, as in-app purchases. Gifts received by a TikTok account are converted into so-called *diamonds* that broadcasters can later cash out to PayPal accounts, but only after having reached a threshold amount, which at the time of this writing is $US 100 dollars [18]. In Section 5, we analyze the forensic artifacts related to TikTok's virtual currency and gift donation.

At the interaction level, TikTok allows a user to like videos and comment on them. Comments can be text or videos. Obviously, a user can read/watch existing comments. By default, the video owner receives notifications regarding new likes and new comments on his/her videos. The video owner can disable comments for a given video. A user can follow another user, receiving a notification when new videos are posted in the account being followed. The top-level of interaction in TikTok is *friendship*. In TikTok parlance, friendship occurs when two accounts follow each other. Friends in TikTok can exchange messages and thus communicate, although sending messages can only be done after the phone number has been linked with the sender's TikTok account.

TikTok enables two main levels of privacy: *i*) video-level and *ii*) account-level. At the video level, the account owner can, at any time, set any of his/her own published videos as private, effectively forbidding others to interact – watch, comment, like – the video. Account-level privacy has a much wider effect, as only users approved by the owner account can interact with the account. This means that solely authorized users can follow the account, watch/like/download posted videos, interact with live performances

Table 3: Hardware and software employed in the study

Name	version	Info
Samsung A40/4 GiB RAM	Android 10.0	Rooted
`android-tools-adb`	1:7.0.0	Smartphone access
`SerializationDumper`	1.13	Java serialization
GNU utilities	various	command-line utilities in Linux
`json.tool`	Python 3 module	JSON Beautifier
`VLC`	3.0.12	Media player

and view the account information. In short, this effectively locks the account content to unauthorized users. Since January 2021, accounts of TikTok users in the age range 13-15 years old are set as private by default [32].

TikTok mobile application provides several features to assist and facilitate the creation of videos with the smartphone. Examples of these features include templates, filters, visual effects coupled to a vast library of music [13].

4 Forensic Artifacts

In this section, we analyze the main forensic artifacts of the Android TikTok app. First, we present materials and methods used to analyze the digital forensic artifacts. We then describe the forensic artifacts collected from the smartphone by first looking at the public data of the TikTok app and then at the private data of TikTok, that is, data that are solely accessible on rooted Android phones.

4.1 Materials and methods

Our study used a rooted Android 10 smartphone, allowing us to access all data of the device, public and private. TikTok version 18.1.3 – the most recent version at the time of the study – was installed. The content of the smartphone was extracted with the Android Debug Bridge (ADB). To analyze the SQLite3 databases present in TikTok, we resorted to Db Browser for SQLite open-source software. Several command-line GNU's utilities such as `file`, `find` and `grep` served to identify binary content and search files. For easier interpretation, JSON files were reformatted with Python 3's `json.tool` module, which is a JSON beautifier. Finally, the tool `SerializationDumper`[3] allowed the extraction and decoding of data from Java serialization-formatted files kept by TikTok. Names and versions of the tools are listed in Table 3.

4.2 Public data

The public data of TikTok are located in a tree hierarchy shown in Figure 3. The entry directory is `/sdcard/Android/data/com.zhiliaoapp.musically`. It is interesting to note that the naming still reflects on the merging with `musical.ly`. We label these data as *public* as they can be obtained through ADB without root access privileges.

Forensically speaking, the most relevant data are within the subdirectories *i*) `picture`, *ii*) `prefs` and *iii*) `awemeCache`. The `picture` directory is part of the open-source Fresco library[4]. Fresco defines

[3] `https://github.com/NickstaDB/SerializationDumper`
[4] `https://frescolib.org`

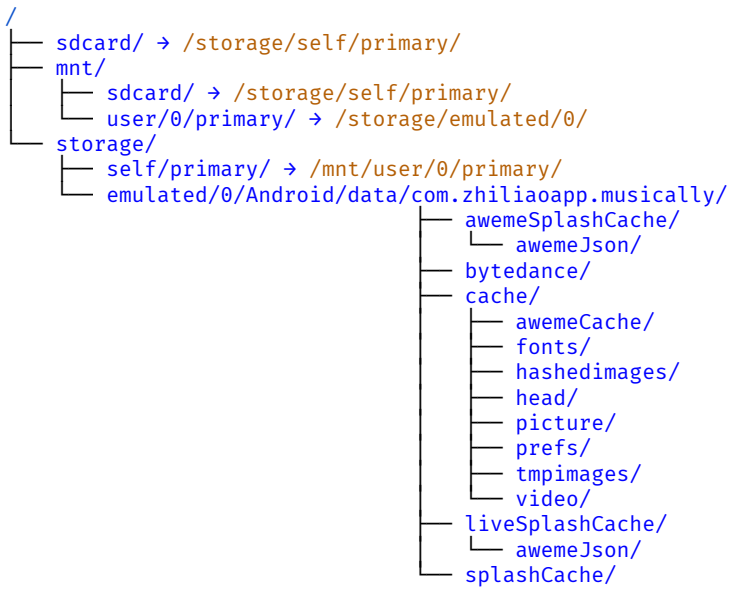


Figure 3: TikTok's app public storage directory hierarchy, including four symbolic links that can be used as alternative paths to `/storage/emulated/0/` and, therefore, reach the public storage. Directories names end with "/", while links are marked with "→" followed by the target directory.

Listing 2: Small extract of `local_prefs.json` file

```
{ "https://imapi-mu.isnssdk.com": {"supports_spdy": true}},
{"https://log16-normal-c-useast1a.tiktokv.com": {"supports_spdy": true}}
```

itself as an image management library that complements Android's image functionalities. The `pictures` directory has a further set of directories, numerically named, from 0 to 99. Inside, each of these directories holds image files, either JPEG or WebP. These files are images loaded during TikTok app usage and correspond to profile pictures and banners from the videos.

The `prefs` directory has a single JSON-formatted file, named `local_prefs.json`. The file mostly holds URL addresses related to TikTok APIs. Listing 2 shows a small extract of the `local_prefs.json` file. Besides the `spdy` reference seen in Listing 2, and which corresponds to `HTTP/2`, the file has also references to the `QUIC` protocol (`HTTP/3`) and to the `HTTPS` interface of Google's public DNS service, `dns.google.com`.

Finally, `awemeCache` holds a set of files whose names correspond to TikTok video identifiers, such as 6912215720931757317. Each file is identified by the Unix `file` command as `Java serialization data, version 5`, and in fact the files can be decoded with the `SerializationDumper` tool. Within the decoded data of a given file, one can found an URL of the form `https://m.tiktok.com/v/videoID.html`, where `videoID` corresponds simultaneously to the file name and the ID of a TikTok video.

Listing 3: Another extract of `local_prefs.json` file

```
"network_qualities": {
    "CAISCyJXaXJlZFNTSUQiGAM=": "4G",
    "CAISCyJXaXJlZFNTSUQiGAQ=": "4G",
    "CAYSABiAgICA+P////8B": "Offline"
}
```

Location	Type	Description
cache/awemeCache	JAVA serialization 5	Info about videos
cache/picture	JPEG/WebP	photos and images
cache/prefs	JSON	DNS, HTTP protocols

Table 4: Most relevant sources from TikTok app's public data

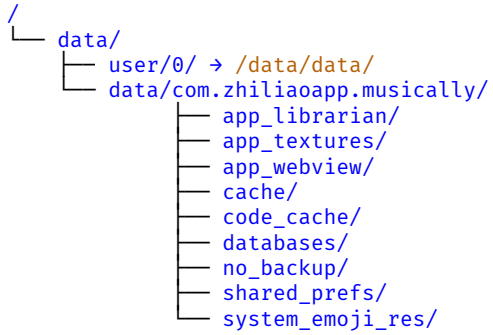


Figure 4: TikTok's app private storage directory hierarchy for Android's main user, including a link that can be used as alternative path.

For example, `https://m.tiktok.com/v/6912215720931757317.html`. Note that the file itself does not contain the video, instead it holds several metadata about the video and the account of the owner.

A summary of the most relevant forensic data of the public part of TikTok storage is presented in Table 4.

4.3 Private data

Private data are only accessible on rooted Android phones and corresponds to the app private storage that is kept in `/data/data/com.zhiliaoapp.musically`. The directory hierarchy, shown in Figure 4, follows Android's rules regarding directory naming for the private storage of applications. The most interesting data from a digital forensic point of view are kept in three directories: `databases`, `cache` and `shared_prefs`. We now focus on each of these directories.

4.4 Databases

In the `databases` directory, there are 30 SQLite 3 databases. Another one – `cookies` – is located in the `app_webview` directory. The names shown in Table 5 correspond to databases that have relevant data for digital forensic examinations, as well as their respective `user\spaceversion` (when it is defined). The remaining datababes are listed in Table 6. Note that the database `<userID>` seen in the first entry of Table 5 – `<userID>_im.db` – needs to be replaced by the account ID that is configured in the app, for example, `6912191700166542342_im.db` for the `nature2admire` account. Moreover, there are as many `<userID>_im.db` databases as there are TikTok accounts configured in the Android app. Next, we summarily describe the most relevant databases for forensic practitioners.

Filename	User version (SQLite version)
<userID>_im.db	21
Cookies.db	–
ss_app_log.db	10
db_im_xx	18
lib_log_queue.db	1
video.db	3
TIKTOK.db	1

Table 5: Forensically relevant SQLite 3 databases of TikTok existing in the database directory. Note that there is one <userID>_im.db file per configured TikTok account.

Filenames	
androidx.work.workdb	google_app_measurement_local.db
apm_monitor_t1.db	i18n_live
aweme.db	npth_log.db
cms_log.db	OriginalSound
db_dynamic_detection_xx	psdkmon_v2.db
DeviceRegisterMonitor.db	runtimeBehavior
downloader.db	share.db
feedback.db	showAd.db
free_flow	splashsdk.db
gecko_clean_statistic.db	ss_push_log.db
gecko_local_info.db	storage_db
google_app_measurement.db	web_offline_statistic.db

Table 6: SQLite 3 databases of TikTok existing in the database directory that do not hold forensic value.

4.4.1 Cookies.db

The Cookies.db database has only one meaningful table, named Cookies. This table holds the session cookies of the app access to TikTok's servers. The date/time fields of the table – creation_utc, expires_utc and last_access_utc – are in a slightly modified format of the Microsoft Filetime 64 since they represent the number of tenths of nanoseconds and not hundredths of nanoseconds as the original Microsoft format. The fields creation_utc and last_access_utc are interesting since they correspond to access by the app to TikTok's social network. Examples of domains whose cookies are kept in the Cookies table are byteoversea.com and musical.ly.

4.4.2 ss_app_log.db

This database logs user's interaction with the app, collecting analytics and monitoring data (*e. g.* network speed). It has has seven tables: event, misc_log, mon_log, page, queue, session and succ_rate. The event table contains entries tagged with names such as click, enter_page, video_request, and video_play_end, with each entry holding in the field ext_json, a JSON string that carries data of the event. The page table logs the user's navigation within the views of the app. Data in the database

Field	Datatype	Description
msg_uuid	text	Unique message identifier
msg_server_id	bigint	ID of message server
conversation_id	text	0:1:{p1}:{p2}, p1=TikTokID1, p2=TikTokID2
type	integer	message type
deleted	integer	0=deleted msg, 1=non-deleted
created_time	datetime	message creation timestamp
status	integer	status of the message
sender	bigint	sender's tikTokID
content	text	message content in JSON format
read_status	integer	0=unread message, 1=read message

Table 7: Main fields of msg table from UserID database [12]

are organized around a common session_id, with the session descriptor kept in session table. As reported by Robert [26], data of the database are part of the periodic analytics sent in JSON-formatted messages to TikTok servers. Data are deleted, possibly after being sent, as the amount of data kept in the database is reduced and only covers a short timespan. However, the rollback file of the database – ss_app_log.db-journal – holds data from the last performed transactions, providing meaningful forensic data, as we shall see in Section 5.

4.4.3 userID_im.db

As stated earlier, there is one <userID>_im.db database per TikTok account configured in the smartphone, with userID corresponding to the account ID. Moreover, the database file persists even after the user logs out of the account. This is expected, as the user might want, later on, to log in again into the account. The userID_im.db has great value for digital forensics, as it holds the messages exchanged between the account linked to the database and other TikTok users. This database contains 14 tables.

Data regarding conversations are kept in three tables: conversation_core, conversation_list and conversation_settings, while messages are kept in the msg table. Specifically, msg table holds the messages exchanged between userID and other TikTok's users. The main fields of msg table are shown in Table 7. Each entry in the table represents an exchanged message. Deleted messages are kept in the table, with the field deleted set to 1. Likewise, the table keeps track whether a message was read or not, with the field read_status set to 1 for unread messages. Furthermore, data in msg allow to retrace messages exchanged between two TikTok users, each of them identified by the text field conversation_id that has the following format: 0:1:ID1:ID2, where ID1 and ID2 are the TikTok IDs of the users engaged in the message, while the field sender keeps the TikTok ID of the message's sender. The field type indicates the type of the message through an integer field. Table 8 lists the values and their meanings that were identified in our testing environment for this field. A msg record has a self explanatory created_time, formatted as an Unix Epoch timestamp. Interestingly, two other fields – index_in_conversation and order_index – use Unix Epoch values. Both resort to Unix timestamp up to milliseconds, but the latter also adds three index digits to the right (*e. g.* 001, 002,...). This way, the value is not only a timestamp but also an index.

Value	Meaning
5	animated gif
7	text
8	video
15	animated gif located in a remote server (*e. g.*, giphy.com or tenor.com)
19	for hashtag
22	audio
25	profile

Table 8: Meaning of the values inside the `type` field

4.4.4 db_im_xx

The database `db_im_xx` encompasses four tables, but only the table `SIMPLE_USER` provides meaningful data. This table has one record per user with whom `userID` has interacted. Each user is identified through his/her respective TikTok account ID in the field `UID`. Besides `UID`, the most relevant data are the `NICK_NAME`, the `AVATAR_THUMB` which holds JSON-formated content of the user's avatar including an URL to the thumbnail, `UNIQUE_ID` which represents the name handle, and `FOLLOW_STATUS`. The `FOLLOW_STATUS` field stores the relationship between `userID` and the listed user, as follows: 0 does not follow but can be followed by `userID`; 1 follows; 2 follows and is followed, that is a *friend* in TikTok's parlance.

4.4.5 lib_log_queue.db

This database has only one relevant table: `queue`. It is associated with the monitoring service. Although in our experiments, the table was empty, we were able to recover several JSON strings that contained monitoring data through the linked journal file `lib_log_queue.db-journal`. These strings held details about the app (version, SDK version, supported HTTP protocols), the device (ID, screen resolution, timezone, model, ...), the network ("WIFI", "4G", download speed, ...), to name just a few. The JSON data also encompassed some TikTok's internal ID such as the already known `userID`, but also `device_id` and `install_id`, among other data. Many of these data correspond to the ones that were identified by Robert as being periodically sent by the app to TikTok's servers [25, 26].

4.4.6 video.db

The `video.db` database logs the HTTPS interaction between the app and TikTok's video repositories, as the name of the only table of the database expresses: `video_http_header_t`. Each row of the table keeps track of a video. One of the fields of the table is `key`, which holds unique values represented in 32-character hexadecimal (128 bits, *e. g.* `A76D7A943A1185919B8DAD308CC918BB`), that might be an MD5 checksum. The name kept in the `key` field is also used as the filename of the video if it exists in the video cache described in Section 4.2.

Other relevant fields are 1) `mime` which represents the MIME type of the video (*e. g.*, "video/ MP4"), 2) `contentLength` which holds the size in bytes of the video, and 3) `extra` which is a JSON-formatted string, itself with three fields: `requestUrl`, `requestHeaders` and `responseHeaders`. These three fields correspond to traditional HTTP protocol elements, respectively, the *URL*, the *request header* and the *response header*. The content *response header* can provide some useful data, namely the UTC-based date/time of the HTTP server that has sent the response. Although the URL is not accessible outside of

the app, as the URL requires proper authentication, the video might still be present in the video cache of the app. This can be assessed by checking, in the video cache directory, for the existence of a file whose name matches the `key` field.

4.4.7 TIKTOK.db

This database holds the table `app_open`, which has only a single field: `open_time`. This field records, in a Unix Epoch format, the date/time whenever the TikTok app is launched.

4.5 The `cache` directory

The `cache` directory hosts two cache-named subdirectories: `cache` and `cachev2`. Contrary to what we observed in an earlier version of TikTok [12], where both directories acted as cache of videos, in this version, the `cache` directory is empty. This suggests that the app has fully transitioned to `cachev2`.

4.5.1 cacheV2

The `cachev2` directory stores videos in files whose names have `mdl` extension. Videos players such as Windows Media Player and VLC fail to decode the files, reporting a format error. The name of certain files such as `v09044a30000_h264_540p_310461.mdl` and `v09044190000bre_h265_720p_945952.mdl` suggests that they are H264 and H265 encoded video files [17]. This is confirmed by inspecting the files, as they contain markers of the MP4 container format (*e. g.*, `mp41` for H264 and `mp42` for H265) and structures of the respective codec (e.g., *avc* for H264 and *hvc1* for H265) [15]. For some of the `mdl` files, there are files with the same name, but with extension `mdlnodeconf` in place of `mdl`. These `mdlnodeconf` files are no larger than 400 bytes and hold content compatible with MP4/H264 headers, namely the `isomiso2mp41` descriptor [15]. Oddly, some of the `mdlnodeconf` exist but are empty (0 bytes), while some `mdl` files have no matching `mdlnodeconf` file.

Nonetheless, the files lack a proper header and thus cannot be watched in a video player, unless the header is fixed. A quick workaround is to overwrite the leading zero bytes of the `mdl` files with an MP4 file header, for either an H264 or H265 video, depending on the format of the `mdl` file. Figure 5 shows the 64 bytes used to *patch* an H265 video.

```
Offset(h) 0001 0203 0405 0607 0809 0A0B 0C0D 0E0F  Decoded text
00000000: 0000 001c 6674 7970 6973 6f6d 0000 0200  ....ftypisom....
00000010: 6973 6f6d 6973 6f32 6d70 3431 0000 6ffa  isomiso2mp41..o.
00000020: 6d6f 6f76 0000 006c 6d76 6864 0000 0000  moov...lmvhd....
00000030: 0000 0000 0000 0000 0000 03e8 0000 8127  ...............'
```

Figure 5: 64-byte patch to apply to a `mdl` CacheV2 file

4.5.2 Other caches

The `cache` directory host two more directories with cache functions: `awemeCache` and `feedCache`. The former has the same content that the same-named directory that exists in the `publicdata` (Section 4.2). The `feedCache` directory holds ready to play, that is, properly formatted MP4 files, whose name corresponds to TikTok videoID (*e. g.* `6912215720931757317`). As the name of the directory suggests, these files are part of TikTok's feed and are the next in line to be shown to the user.

Filename	Main forensic interests
applog_stats.xml	timestamps; WiFi BSSID (old versions of TikTok)
aweme_user.xml	email address and phone number (both partial)
LoginSharePreferences.xml	full email address or full phone number (depends on the login)
push_multi_process_config.xml	several TikTok identifiers
search.xml	recent searches (strings entered)
wschannel_multi_process_config.xml	device (data, configuration)

Table 9: Most relevant XML files

4.6 XML files of shared_prefs

Besides SQLite databases, TikTok app also uses a large number of XML files. In the studied version of TikTok version, there are 110 XML files, all located in the `shared_prefs` directory. Table 9 lists the most relevant XML files and their potential contribution to a digital forensic examination. Next, we briefly review each of these forensically relevant XML files.

4.6.1 applog_stats.xml

This file holds some data associated to logging. Data include some TikTok's identifiers (`device_id` and `install_id`), plus several date/time related fields, such as `send_fingerprint_time`, `last_config_time` and `app_log_last_config_time` All date/time fields are expressed in Unix Epoch milliseconds format. The file also references several addresses of the `pstatp.com` domain (*e. g.*, `p5.pstatp.com`).

Regarding information of the network, we found two entries with network identifiers, namely BSSID and SSID: `npth_privacy_detection_dynamic_wifi_bssid_error` and `npth_privacy_detection_dynamic_wifi_ssid_error`, both followed with the value 1. Intrigued by the presence of these entries, we analyzed a dump of the same smartphone made when it had TikTok version 16.0.41 [12]. We found out that the file `applog_stats.xml` the older version held two entries related to the BSSID (Basic Service Set IDentifiers): `last_wifi_bssid` and `last_check_bssid_time`. The first entry held the correct BSSID of the WiFi access point, while the latter held, again in Unix Epoch milliseconds format, the last date/time the BSSID was collected. Moreover, when we updated TikTok from version 16.0.41 to 18.1.3, the XML file kept these two entries, with the same values, adding the two `_error` ones. While the entries `last_wifi_bssid` and `last_check_bssid_time` still exist when TikTok is upgraded from a previous version, a fresh installation of TikTok confirms that the BSSID is no longer collected by the TikTok app. Note that the practice of collecting network identifiers has been heavily criticized in the past [23].

4.6.2 aweme_user.xml

This file holds data regarding TikTok's accounts which are configured in the smartphone. To preserve space, a shortened list of main data is given in Table 10. Note that contrary to previous versions [12], the file no longer keeps the whole email address, instead only the first and last letter of the username plus the full domain name are kept in the file. Moreover, only four digits plus the international phone code are kept, and, obviously, only for users that have linked their phone number to TikTok. The file contains a set of entries per each user whose account is configured in the application. The `aweme_user.xml` file should definitely be consulted in a digital forensic examination. Other relevant fields include whether

Name	Comment
nickname	(self explanatory)
uid	userID
uniqueID	@name
avatar_url	URL to avatar
register_time	timestamp of account creation
country_code	country phone prefix
bind_phone	last 4 digits of phone number (if provided)
email	email address (if provided)
birthday	(if provided)
location	(if provided)
gender	(if provided)
share_url	public URL of account
is_minor	user is minor or not
is_blocked	whether user is blocked by any other account
is_blocking	whether user is blocking by any other account

Table 10: Main fields of `awe_me` XML file [12]

the TikTok account is linked to Facebook, Twitter, Weibo, Youtube, or Instagram, and the number of *followers*, *friends* and of *followed accounts*.

4.6.3 LoginSharePreferences.xml

This file registers data linked to the current session of the user in the TikTok network. Specifically, for an email-based login, that is, the user employed his/her email address to log in TikTok, the file holds the full email address. For a login realized with the phone number, the file keeps the full phone number. Additionally, the file holds in the field `expires` the date/time expiration, in human format (e.g., *Jan 10, 2021 12:49:48*), of the session. The file also keeps the `userID`.

4.6.4 push_multi_process_config.xml

This file has the particularity of holding all TikTok identifiers, namely: `device_id`, `user_id`, `install_id`, and `clientudid`. This can be useful to cross reference data in other XML files and databases.

4.6.5 search.xml

As the name suggests, the XML file `search.xml` keeps track of the recent searches performed by the app's user. The history of recent searches are kept in the XML file under the entity `recent_history_v2`. Note that the XML file solely holds the string searched: no results and no timestamps are kept, although each entry identifies the account ID that performed the search.

4.6.6 wschannel_multi_process_config.xml

This file encompasses, in a single entry, the `device_id` and `install_id` with data about the device and configuration, namely, device type, OS type, OS version, region, access (e.g., WiFi, 4G, *etc*), timezone,

screen resolution, etc.

5 Forensic artifacts of Tiktok's virtual currency

TikTok supports digital rewards, by which a Tiktok user can donate digital gifts to a TikTok user who is performing a live session. Digital gifts need to be purchased with TikTok in-app currency, which is called *TikTok coins* (henceforth, coins). In this section, we analyze the digital forensic artifacts left when operations involving TikTok's coins and digital gifts are performed. The analyzed operations are *i*) purchasing; *ii*) gifting; and *iii*) receiving. For this, the experiment methodology was as follows: the coin operations were performed from the rooted smartphone, and after each operation, an ADB dump was collected from the smartphone. Note that in this section, to preserve privacy, we have changed TikTok's ID shown in listings, replacing 14 (out of 19) of their inner digits with zeros.

5.1 Purchasing coins

Coin purchase is done through the mobile application by accessing the `recharge` option of the `balance` menu. Coins can be bought in amounts of 70, 350, 1 400, 3 500, or 7 000. The transaction occurs between the user and Google Pay services. Since December 2019, the minimum age for buying coins is 18 years old [30].

For this experiment, 70 coins were purchased at 1.09 Euros. The ADB dump collected soon after the transaction revealed traces of the coin purchase in two files: `ss_app_log.db-journal` and `psdkmon_v2.db-journal`. Both contents are JSON-formatted text, see Listing 4 and Listing 5. To preserve privacy, TikTok IDs shown in both listings have their inner digits replaced with zeros.

Listing 4 shows the extract of the `psdkmon_v2.db-journal` file that corresponds to the purchase, as all three timestamps – `timestamp`, `local_time_ms`, and `datetime` – match the local date/time of the transaction. Both `timestamp` and `local_time_ms` are in Unix EPOCH milliseconds format, while `datetime` is in human readable format. The `request_id` element encompasses in its last 19 digits a TikTok identifier. When decoded, the highest 32-bit of TikTok's identifier that lies within the `request_id` holds the timestamp of the purchase, while the lowest 32-bits form a `B` in the third least significant hexadecimal symbol.

To preserve space, Listing 5 only shows the forensically relevant content found in `ss_app_log.db-journal`. Data shown in the listing include the purchaser's TikTok identifier (`user_id`), the amount of acquired coins (`recharge_package`, 70 in this case), the pay method (`pay_method`) and the local date/time in `local_time_ms` and `datetime`, both having same formats as the equally named fields of Listing 4. Note that the `pay_method` mentions "Google Pay": in-apps purchases are performed through Google Play services, with the buyer receiving, through email, an invoice documenting the transaction.

Interestingly, based on the value of their respective `event_id` – 11825 for Listing 4 and 11826 for Listing 5, and although coming from different files, the listings show two consecutive events. Furthermore, as observed in Section 4.4.2, the `ss_app_log.db` database logs application's events. Thus, the existence of coins' purchase data in the journal file of the `ss_app_log.db` database, that is, `ss_app_log.db-journal` means that purchase transactions are also part of the logged events.

From a digital forensic standpoint, a major issue with the artifacts found from the purchase transaction is their locations: artifacts are in `-journal` files of SQLite 3 databases. A `-journal` file is a *rollback journal*, which provides support for atomic commits and rollback operations. It holds the undo log, that is old database pages [33]. The content of a `-journal` file is short-lived, meaning that the possibility of finding artifacts is severely reduced unless the smartphone is examined shortly after the artifact-creating operation has taken place. In fact, when we repeated the ADB dump acquisition after

Listing 4: Partial content of the `psdkmon_v2.db-journal` file, corresponding to the in-app purchase of 70 coins.

```
{
    "nt":4,
    "user_id":6910000000000000021,
    "---omitted for brevity---":...,
    "event":"pay_callback_event",
    "params":{
        "result_code":0,
        "result_detail_code":0,
        "result_message":"pay success in QueryOrderStateCallback.",
        "pay_type":"NOMAL",
        "product_id":"com.zhiliaoapp.musical.iap.coins.v2.100",
        "request_id":"10000016940000000000000065",
        "user_id":"6910000000000000021",
        "timestamp":1617822494000
    },
    "event_id":11825,
    "tea_event_index":70,
    "local_time_ms":1617822494000,
    "session_id":"00000000-0000-0000-0000-000000000000",
    "datetime":"2021-04-07 20:08:14"
}
```

Listing 5: Partial content of the `ss_app_log.db-journal` file where `recharge_package` correspond to the amount of purchased coins (line 8). Fields `user_id`, `local_time_ms`, and `session_id` were anonymized by replacing original values with zeros.

```
{
   "nt":4,
   "user_id":6910000000000000021,
   "---omitted for brevity---":...,
   "event":"livesdk_recharge_pay",
   "params":{
      "recharge_package":"70",
      "request_page":"my_profile",
      "pay_method":"Google Pay"
   },
   "event_id":11821,
   "tea_event_index":66,
   "local_time_ms":1617822466000,
   "session_id":"00000000-0000-0000-0000-000000000000",
   "datetime":"2021-04-07 20:07:46"
}
```

some basic TikTok usage, the purchase artifacts were no longer found.

5.2 Gifting coins

When watching a live TikTok session, a TikTok user can donate virtual gifts to reward the live session performer. The gifts are graphical icons, which briefly appear moving up in the video feed of the live session. Each gift costs a given amount of TikTok coins. Cost ranges from 1 coin for a `Rose` to 5 000 coins for the `DramaQueen`. Note that 5 000 coins cost, in April 2021, roughly 78 euros. Table 11 lists some of the available gifts and their internal TikTok ID. Within TikTok, a more complete list of gifts exists in the `shared_prefs/live_sdk_core.xml` file.

We found artifacts of coin gifting in the `ss_app_log.db-journal` file, linked to a JSON-formatted log entry for the `livesdk_gift_send_click` event. A trimmed view of the entry is shown in Listing 6. Forensically speaking, the most relevant elements are: *i*) `user_id` which corresponds to the TikTok account ID that donated the gift; *ii*) `room_id` which is the ID of the live session where the gift donation

Key	Internal ID	Coins	Name
live_gift_5655	5655	1	Rose
live_gift_5269	5269	1	TikTok
live_gift_5650	5650	5	Mic
live_gift_5656	5656	7	Flowers
live_gift_5651	5651	1 500	Garland
live_gift_5489	5489	2 020	Carousel
live_gift_5652	5652	3 000	Ferris Wheel
live_gift_5232	5232	5 000	Drama Queen

Table 11: Some examples of gifts' names, IDs and cost in TikTok coins.

Listing 6: Extract of the `databases/ss_app_log.db-journal` file. The `livesdk_gift_send_click` event registers gifts sent to another user. The `gift_id` is 5655, which corresponds to a `Rose`, worth 1 TikTok coin. The `anchor_id` identifies the receiving account, while `actual_room_id` is the live session's ID. Fields `user_id`, `room_id`, `anchor_id`, and `session_id` were anonymized with zeros.


```
{
   "nt":4,
   "user_id":6910000000000000021,
   "---omitted for brevity---":...,
   "event":"livesdk_gift_send_click",
   "params":{
      "room_id":"6950000000000000038",
      "live_window_mode":"live_big_picture",
      "action_type":"click",
      "---omitted for brevity---":...,
      "live_type":"video_live",
      "---omitted for brevity---":...,
      "anchor_id":"6860000000000000065",
      "gift_id":"5655",
      "---omitted for brevity---":...,
      "actual_room_id":"6950000000000000038",
      "---omitted for brevity---":...,
      "gift_type":"single_gift",
      "---omitted for brevity---":...,
      "gift_cnt":"1"
   },
   "---omitted for brevity---":...,
   "local_time_ms":1618584189000,
   "session_id":"00000000-0000-0000-0000-000000000000",
   "datetime":"2021-04-16 15:43:09"
}
```


took place; *iii*) `anchor_id` which is ID of the live session's performer and thus the TikTok account that receives the gift; and *iv*) `gift_id` that identifies the donated gift, which is 5655 corresponding to a `Rose`. Except for the `gift_id`, all others are TikTok's IDs that follow the 32-bit + 32-bit format.

5.3 Receiving coins

A TikTok user can only receive gifts when he/she is performing a live session. To be able to perform a live session, a TikTok account requires a minimum of 1 000 followers, while any TikTok user can watch and reward the live session, regardless of whether he/she follows the account of the live performer. Received gifts are credited to the live session performer's account, but in another TikTok unit, called *diamonds*. Although the coin-to-diamond conversion is 1 to 1, a diamond is only about one third the value of a coin in fiat money. A TikTok user can cash his/her diamonds when the accumulated diamonds

Listing 7: Extract of the `databases/ss_app_log.db-journal` file that shows the event triggered by the end of a live session performed by `anchor_id`. IDs are filled with zeros for privacy issues.

```
{
    "nt":4,
    "user_id":6910000000000000021,
    "---omitted for brevity---":...,
    "event":"livesdk_live_end_duration",
    "params":{
        "room_id":"6940000000000000078",
        "duration":"31900",
        "anchor_id":"6910000000000000007",
        "room_orientation":"0",
        "follow_status":"0",
        "sdk_version":"1800",
        "live_type":"video_live",
        "---omitted for brevity---":...,
    },
    "event_id":13286,
    "tea_event_index":417,
    "local_time_ms":1617990743000,
    "session_id":"00000000-0000-0000-0000-000000000000",
    "datetime":"2021-04-09 18:52:23"
}
```

are worth 100 USD or plus. The cash-out is performed through a PayPal account. In our experiments, we could not assess forensic evidence of cash-out operations due to the 100 USD threshold needed to trigger a payment.

We did not find evidence about gifts received within a live session. However, we found artifacts left when the user performs a live session. Listing 7 shows the `livesdk_live_end_duration` event, which was found in the `databases/ss_app_log.db-journal` file. The most relevant fields are TikTok's ID `room_id` and `anchor_id`, while `duration` measures the duration of the live session in hundredths of seconds. Specifically, `room_id` identifies the live session, with the highest 32 bits corresponding to the date/time creation of the live session, while `anchor_id` identifies the account that performed the live session.

5.4 Keywords for coin-related artifacts

As seen in the coin-related sections, artifacts related to coin and gift usage – purchase, gifting, and receiving – are scarce or even non-existent. Most of the artifacts were found in `-journal` files, namely in the SQLite 3 `databases/ss_app_log.db-journal` file. However, the high turnaround of data in the `ss_app_log.db` database and its associated journal file [33], means that there is no guaranteed place to look for evidence of coin-related operations. We believe that an appropriate approach for a practitioner to look for coin-related evidence is to search the phone content for specific keywords. This can be done with software able to index and search content, such as Autopsy's keyword search, or through Unix's command line such as `grep`. Table 12 lists the keywords linked to coin-related transactions and their corresponding description.

6 FAMA

The `FAMA` (Forensic Analysis for Mobile Apps) is a modular digital forensic framework. We developed FAMA to help digital practitioners to deal with the numerous mobile applications for Android. Indeed, Google Play hosts millions of Android applications, all of them having their own specificity, namely databases, configuration files, and logs, to name just the more relevant forensic data sources. Even,

keyword	Description
anchor_id	ID of a live session performer
livesdk_gift_send_click	registers a gift sent to another user and its properties
recharge_package	registers the amount of coins bought
room_id	TikTok ID of a live session
pay_callback_event	registers payment messages
pay_method	successful payment, always "Google Pay" in our experiments
<userID1>:<userID2>	interactions between 2 users

Table 12: Keywords to search for on a TikTok forensic analysis about users interaction, coins purchase and gifting.

if a reduced number of apps are really relevant to forensic investigations, developing software, from a zero base, extracting and processing digital forensic artifacts is cumbersome and time-consuming. The goals of FAMA is to facilitate both *i*) the extraction of data from Android smartphone and *ii*) the forensic analysis of the extracted data. For this purpose, FAMA supports data collection via ADB and the execution of tailored scripts for apps. To ease the development of those scripts, FAMA provides a framework with several coding facilities to interact with SQLite3 databases, XML files, geolocation coordinates, multimedia files, and so on. FAMA also supports the creation of a timeline, a handy tool for forensic analysis.

When faced with a yet not supported app, one can quickly develop support for the app within FAMA. FAMA has been developed by our team in Python, runs in the three major platforms – Windows, Linux, and macOS – and is available as open source [21].

Regarding data extraction, FAMA can work with non-rooted Android phones, although extracted data will be limited to the publicly available, thus missing important elements, as shown earlier for TikTok.

6.1 Integration with Autopsy

The Autopsy[5] software is a well-known and reputed open-source software for digital forensics. It allows digital forensic practitioners to perform analysis within an intuitive graphical interface. Specifically, the practitioner creates a case in Autopsy, adding the data sources that he/she pretends to examine. Then, the practitioner can apply several Autopsy's tools and modules, with the results being integrated into Autopsy's internal structures, and shown within the tool graphical interface. As tools are integrated into Autopsy, the practitioner does not have to handle the details, even for external tools such as PhotoRec for file recovery and carving, Tika for file format detection, SOLR as the content indexer and search engine, and RegRipper to analyze and extract data from the Windows OS registry, just to name a few [29]. Autopsy also provides for the automated creation of reports, a useful functionality for forensic practitioners.

Autopsy can be extended with software modules. These extension modules need to be coded either in Java or in Jython. Jython is a version of Python that runs within the context of a Java Virtual Machine. Moreover, Jython is currently limited to Python 2.x. More importantly, an Autopsy module must be coded practically from zero and the interaction with Autopsy data interface – BlackBoard – is not trivial. Autopsy also provides dedicated support for Android through its Android Analyzer environment, which allows for the development of modules to extract and/or analyze from Android forensic images. To develop a python module for Autopsy's Android Analyzer, one has to follow the structure of An-

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

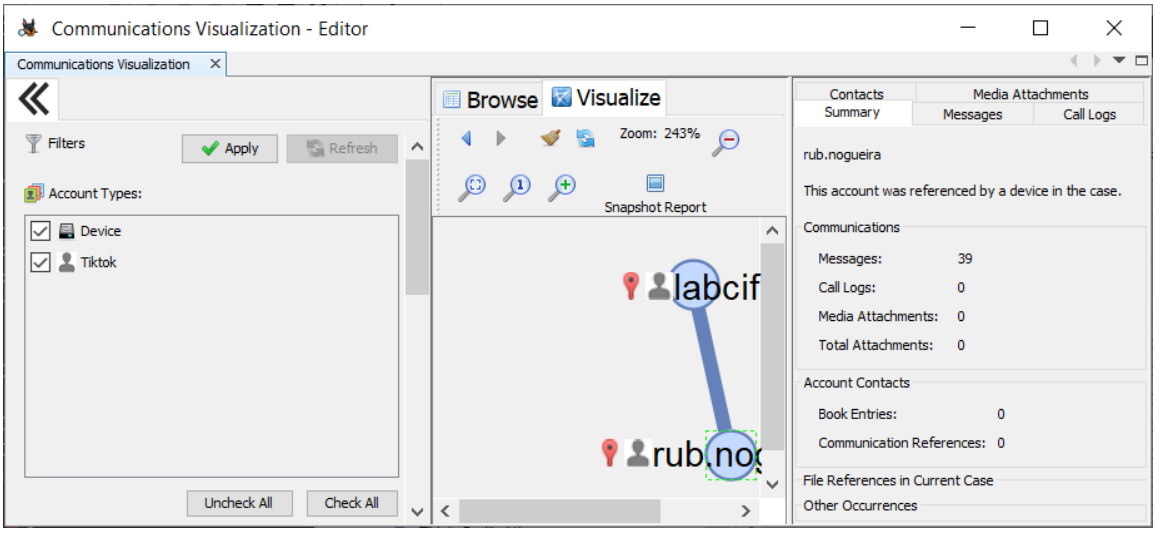


Figure 6: Messages exchanged within the TikTok network between two accounts

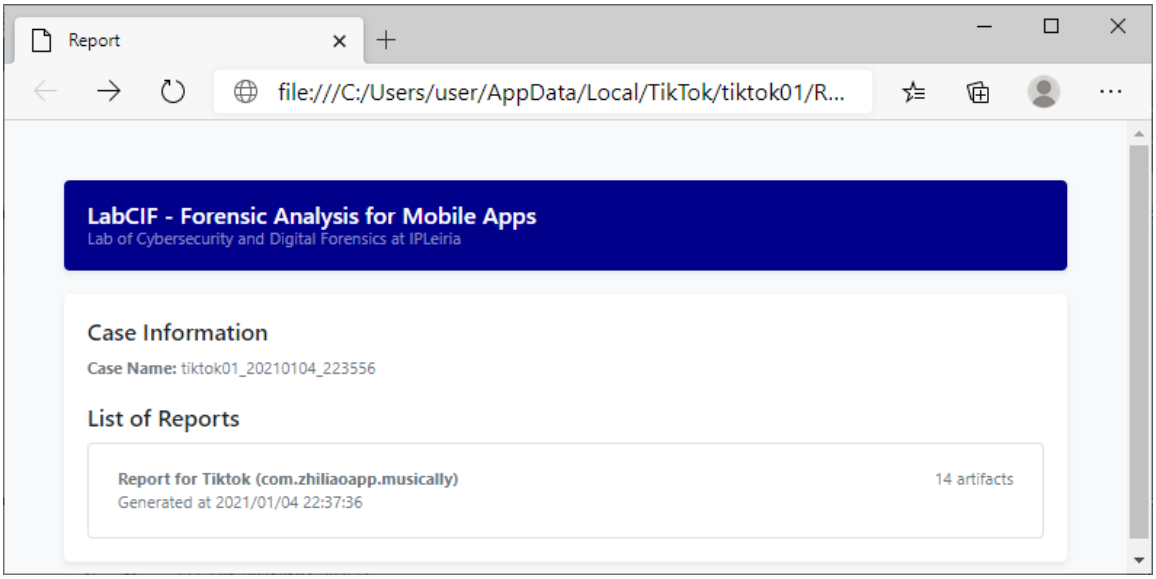


Figure 7: FAMA front end for TikTok HTML report

droid Analyzer, implementing the required classes and methods. However, functionalities are limited as we experimented in previous work when we developed a TikTok module fitted to Autopsy's Android Analyzer [12].

FAMA provides code models and a set of Python classes that smoothen the development of FAMA's modules, based on our experience of developing Autopsy and Android Analyzer's modules. Moreover, while FAMA can be run within a regular terminal, FAMA can also be integrated within Autopsy through a module. For this purpose, the framework offers three main components: *i*) Data source processor, *ii*) Ingest, and *iii*) Report. Details about FAMA are available at FAMA's GitHub repository[6].

[6]`https://github.com/labcif/FAMA`

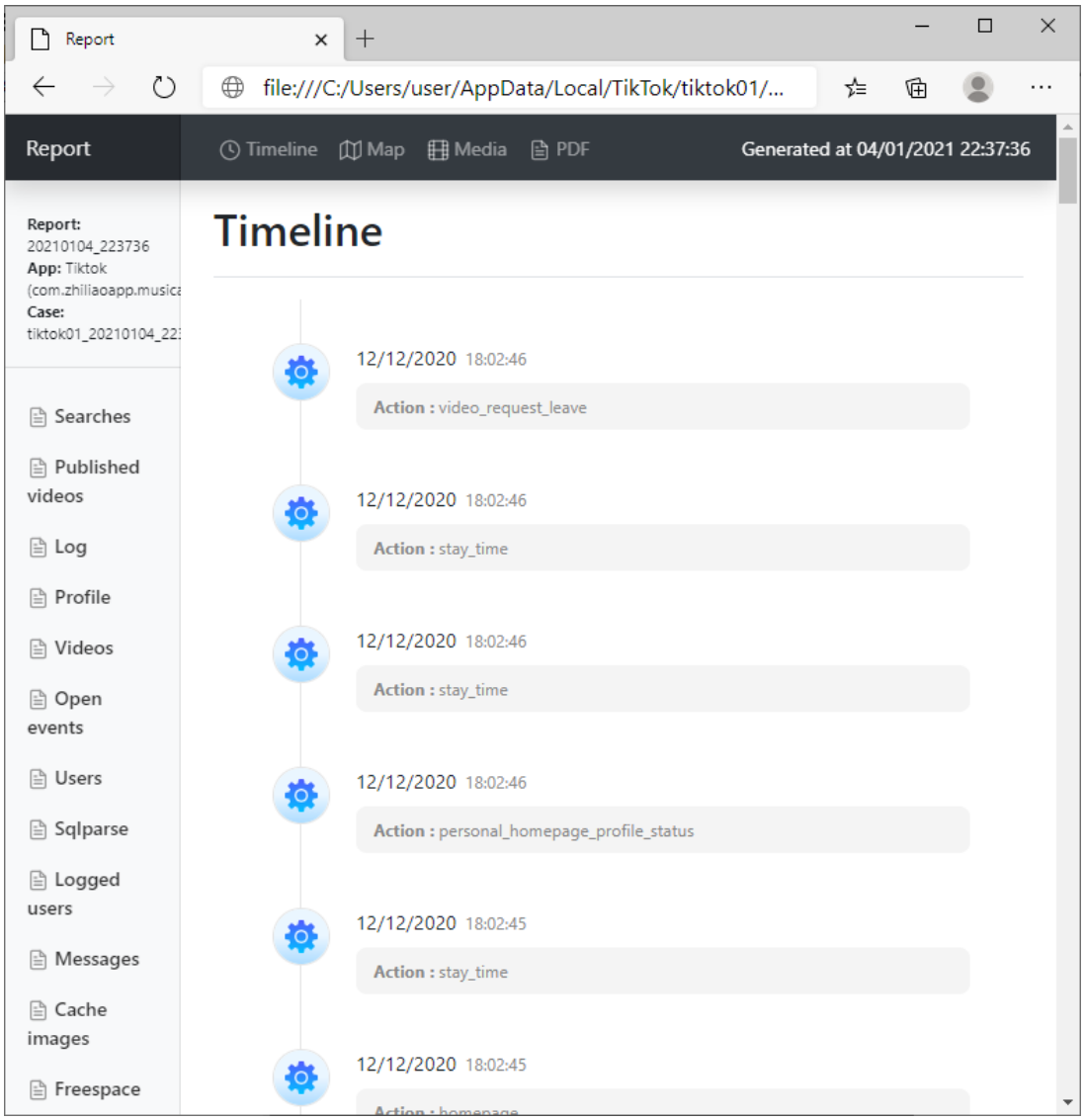


Figure 8: FAMA core view for TikTok HTML report

6.2 Support for TikTok

Support for TikTok within Autopsy/FAMA allows ingesting the smartphone data related to TikTok. As stated earlier, FAMA can collect data from an Android smartphone acting as a simple interface to ADB. Moreover, based on the AppID, FAMA can selectively collect the data of the Android app in the connected smartphone. This can be done directly from the terminal, or through Autopsy, although this second options presently requires a patch[7] to Autopsy.

FAMA generates two `tar.gz` files. One called VERSION.external.tar.gz, and another one called VERSION.internal.tar.gz. `VERSION` represents the Android app version, while external and internal holds, respectively, the publicly/privately accessible data of the app. As stated earlier, the creation of the internal VERSION.tar.gz archive requires a rooted smartphone.

These archives are then fed to Autopsy as data sources, and FAMA ingest module is then run. On

[7] `https://github.com/sleuthkit/autopsy/pull/6027`

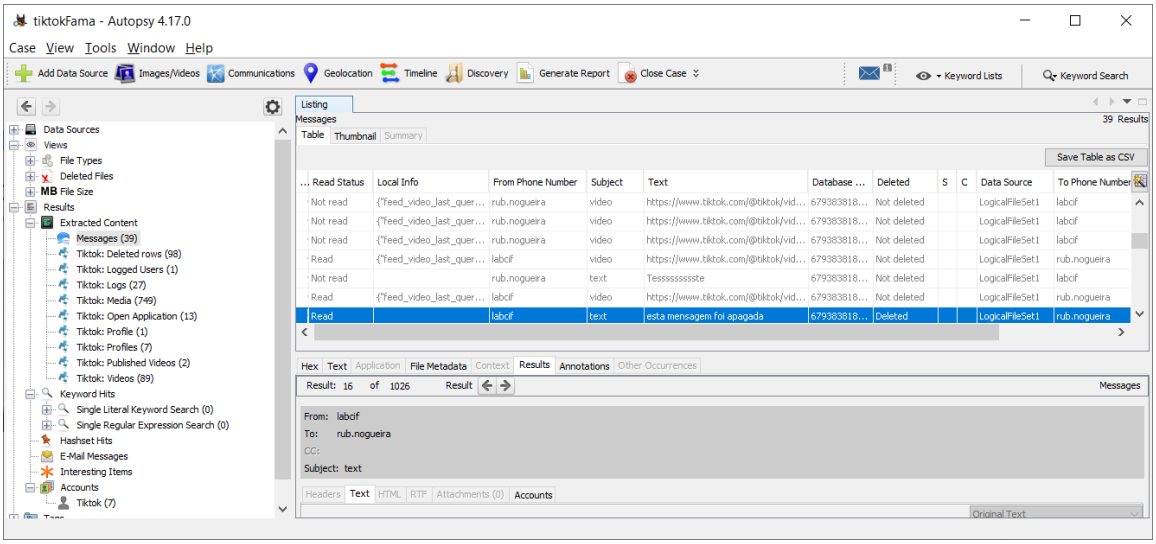


Figure 9: TikTok data in Autopsy after FAMA's ingest process

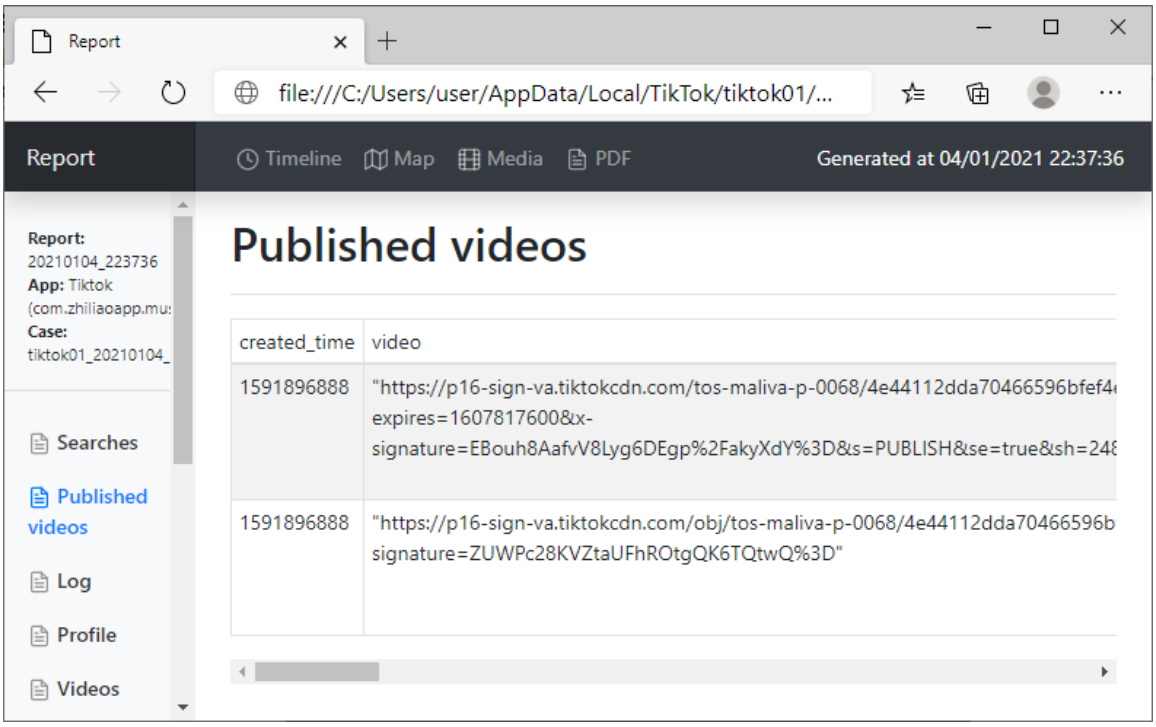


Figure 10: FAMA report showing analyzed TikTok's account list of Published videos

LabCIF - Forensic Analysis for Mobile Apps Report
Tiktok - Report 20210104_223736

Case Name: tiktok01_20210104_223556
Category: Searches
Category: Published_videos

created_time	video	api_address
1591896888	"https://p16-sign-va.tiktokcdn.com/tos-maliva-p-0068/4e44112dda70466596bfef4e01a1445d_1591896890~tplv-tiktokx-cropcenter:248:330.awebp?x-expires=1607817600&x-signature=EBouh8AafvV8Lyg6DEgp%2FakyXdY%3D&s=PUBLISH&se=true&sh=248_330&sc=dynamic_cover&l=20201212180144010115004060 10AD4A8B"	"https://api-h2.tiktokv.com/aweme/v1/play/?video_id=v09044f70000brh6mcjt7nqtfp3b5cig&line=0&ratio=540p&media_typ
1591896888	"https://p16-sign-va.tiktokcdn.com/obj/tos-maliva-p-0068/4e44112dda70466596bfef4e01a1445d_1591896890?x-expires=1607814000&x-signature=ZUWPc28KVZtaUFhROtgQK6TQtwQ%3D"	"https://api-h2.tiktokv.com/aweme/v1/play/?video_id=v09044f70000brh6mcjt7nqtfp3b5cig&line=0&ratio=540p&media_typ

Category: Log

session_id	body	action	time
123	{"ab_sdk_version":"50017293,50001140,50019350,1472623"}	"personal_homepage_profile_status"	1607796159
123	{"duration":"1820","author_id":"","ab_sdk_version":"50017293,50001140,50019350,1472623","page_uid":""}	"stay_time"	1607796159
123	{"ab_sdk_version":"50017293,50001140,50019350,1472623"}	"livesdk_topview_show_failed"	1607796159
123	{"ab_sdk_version":"50017293,50001140,50019350,1472623"}	"splash_ad"	1607796159
123	{"ab_sdk_version":"50017293,50001140,50019350,1472623"}	"discovery_video_play"	1607796161
123	{"is_first":"0","duration":"0","ab_sdk_version":"50017293,50001140,50019350,1472623"}	"video_request"	1607796161
123	{"is_first":"0","duration":"0","ab_sdk_version":"50017293,50001140,50019350,1472623"}	"video_request"	1607796161

Figure 11: Partial view of FAMA's generated report for TikTok example case

successful execution, data are extracted and processed from the archive(s) and then imported into Autopsy, as shown in Figure 9. As can be observed from Figure 9, extracted and processed data are shown with the `Extracted Content` tree of the Autopsy Interface. The artifacts can be browsed, with details being shown on the right side of Autopsy interface. A note about the entry "TikTok: Deleted rows (98)" visible in the `Extracted Content` tree: it encompasses data recovered from SQLite 3 database resorting to DeGrazia's tool to recover SQLite records [11]. This functionality of recovering SQLite data is integrated within FAMA and can be used by other modules.

Figure 6 displays a graph highlighting the messages exchanged between the analyzed TikTok account and another TikTok account. On the right side, one can see that 39 messages were exchanged between the two accounts. Note that the interface – Communication Editor – is the one made available by Autopsy for easily showing calls and messages (SMS, emails, *etc*) between users in a case. This is the reason why there is an (empty) `Call Logs` entry in the right side of the interface.

FAMA can also generate a set of reports in HTML format. The reports are dynamic, meaning that the user can navigate and access several elements. The front end HTML page for the TikTok module is shown in Figure 7, where it informs that there are 14 artifacts. The core of the HTML report is shown in Figure 8, with the `Timeline` view selected. The possible views – `Timeline`, `Map`, `Media` and `PDF` – are selected at the top of the page. In the vertical left bar, one can select the type of artifacts to be shown in the main window. For example, by selecting `Published videos`, a list of the published videos is displayed as shown in Figure 10. Finally, a very partial and cropped view of the PDF report generated for the TikTok example case through the vertical bar option `PDF` is shown in Figure 11

7 Conclusion

TikTok has quickly become an important social network, installed in several millions of smartphones, with a predominance in the teenage population. It is thus important to study data and information that can be gathered in digital forensic analysis. For this purpose, this work studied TikTok for Android, analyzing both the public and private parts of the mobile application.

The number of SQLite 3 databases – 31 – was surprising, although understandable from a developer perspective, has the app integrated a previous product –`musical.ly` –, and has evolved along 18

versions.

From the digital forensic perspective, most of the relevant data are located in the private storage area of the app, which is only accessible in a rooted smartphone. From this private area, important data for the configured account in the examined smartphone can be gathered. Examples include the last watched videos, list of friends in TikTok network, followers, followed accounts, and the messages exchanged between friends. Also of relevance is the simple yet useful algorithm employed by TikTok to generate most of its 64-bit numerical identifiers, such as `install_id`, `user_id` and `video_id`: the most significant 32 bits corresponds to the date/time timestamp, in Unix format, of the ID creation, and as we have identified, patterns exist in the least significant 32 bits that allows to quickly identify whether a TikTok ID corresponds to an account, a video, a live session and so on. This way, for example, one can easily determine when a video was posted to the network, when an account was created, or when a live session was launched. This can be valuable for a digital forensic investigation.

Regarding personal data, it was possible to extract partially email address or phone number, depending on which one was used to create the account, or for the phone number, whether the user had posted comments on TikTok network, as posting comments requires linking the phone number to the TikTok network. Additionally, for users logged in TikTok with their email address, the full email address is recorded in TikTok's private `LoginSharePreferences.xml` file. The same occurs when the phone number is used as a login identifier. Other data included phone hardware and OS, time zone, network type (WiFi, 4G, etc.) and speed, country, and whether other social network accounts were linked to the account. Our study also looked for artifacts left by interactions with TikTok's coins and digital gifts. Although some artifacts were recovered, none were kept in persistence files, as all artifacts existed in SQLite 3 `-journal` files, which are by definition, ephemeral and short-lived. Nonetheless, we provide a set of coin-related keywords which can be looked up in TikTok's data to detect forensic artifacts.

To ease the extraction and analysis of Android forensic artifacts, we provide the open-source FAMA framework. Coupled with the TikTok module, FAMA analyzes and allows to generate different types of reports on TikTok artifacts that it detects on an Android smartphone.

As future work, we plan to continue to develop and adapt FAMA to the new versions of TikTok. Additionally, we aim to introduce within FAMA the concept of versions, so that a given module will be able to target a precise version of the app. Indeed, from our experience gathered with TikTok, we observed that the app may change quite substantially between versions, at least from the digital forensic point of view. Finally, we also aim to study the interaction of TikTok app with the network.

Acknowledgments

This work was partially supported by CIIC under the FCT/MCTES project UIDB/CEC/4524/2020, and EU funds under the project UIDB/EEA/50008/2020.

References

[1] K. E. Anderson. Getting acquainted with social networks and apps: it is time to talk about TikTok. *Library Hi Tech News*, 37(4):7–12, February 2020.

[2] BBC. India bans TikTok, WeChat and dozens more Chinese apps, June 2020. https://www.bbc.com/news/technology-53225720 [Online; accessed on September 15, 2021].

[3] BBC. TikTok and WeChat: US to ban app downloads in 48 hours, September 2020. https://www.bbc.com/news/technology-54205231 [Online; accessed on September 15, 2021].

[4] BBC. TikTok faces legal action from 12-year-old girl in England, December 2020. https://www.bbc.com/news/technology-55497350 [Online; accessed on September 15, 2021].

[5] R. Benson. Tinkering with TikTok Timestamps, September 2020. https://dfir.pubpub.org/pub/9llea7yp/release/1 [Online; accessed on September 15, 2021].

[6] R. Benson. Tinkering with tiktok timestamps. *DFIR Review*, 9 2020. https://dfir.pubpub.org/pub/9llea7yp.

[7] K. Billups. *New and Emerging Mobile Apps Among Teens-Are Forensic Tools Keeping Up?* PhD thesis, Purdue University Graduate School, May 2020.

[8] C. B. Bossen and R. Kottasz. Uses and gratifications sought by pre-adolescent and adolescent TikTok consumers. *Young Consumers*, 21(4):463–478, November 2020.

[9] A. Brignoni. Finding TikTok messages in Android, December 2018. https://abrignoni.blogspot.com/2018/11/finding-tiktok-messages-in-android.html [Online; accessed on December 14, 2020].

[10] Btf_117. TikTok OSINT: targeted user investigation, April 2020. https://medium.com/@BTF117/tiktok-osint-targeted-user-investigation-9e206f8bb794 [Online; accessed on September 15, 2021].

[11] M. DeGrazia. Sqlite-deleted-records-parser: recovering deleted entries in sqlite database, 2015. https://github.com/mdegrazia/SQLite-Deleted-Records-Parser [Online; accessed on September 15, 2021].

[12] P. Domingues, R. Nogueira, J. C. Francisco, and M. Frade. Post-mortem digital forensic artifacts of tiktok android app. In *Proc. of the 15th International Conference on Availability, Reliability and Security (ARES'20), Ireland*. ACM, August 2020.

[13] J. Herrman. How tiktok is rewriting the world - the new york times, March 2019. https://www.nytimes.com/2019/03/10/style/what-is-tik-tok.html [Online; accessed on September 15, 2021].

[14] N. Hoang Khoa, P. The Duy, H. Do Hoang, D. Thi Thu Hien, and V. Pham. Forensic analysis of tiktok application to seek digital artifacts on android smartphone. In *Proc.of the 2020 RIVF International Conference on Computing and Communication Technologies (RIVF'20), Ho Chi Minh City, Vietnam*, pages 1–5, October 2020.

[15] M. Iuliani, D. Shullani, M. Fontani, S. Meucci, and A. Piva. A video forensic framework for the unsupervised analysis of MP4-like file container. *IEEE Transactions on Information Forensics and Security*, 14(3):635–645, March 2019.

[16] D. B. V. Kaye, X. Chen, and J. Zeng. The co-evolution of two chinese mobile short video apps: Parallel platformization of douyin and TikTok. *Mobile Media & Communication*, pages 1–25, Aug. 2020.

[17] M. Malhotra, A. V. Singh, and R. Matam. Comparative performance issues with h.264 vs h.265. In *Proc. of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMIT-Con'19), Faridabad, India*. IEEE, February 2019.

[18] M. Mhalla, J. Yun, and A. Nasiri. Video-sharing apps business models: Tiktok case study. *International Journal of Innovation and Technology Management*, 17(07):2050050, December 2020.

[19] B. Newsroom. TikTok and WhatsApp top the app downloads for 2019, Janauary 2020. https://www.bbc.co.uk/newsround/51117970 [Online; accessed on September 15, 2021].

[20] A. Neyaz, A. Kumar, S. Krishnan, J. Placker, and Q. Liu. Security, Privacy and Steganographic Analysis of FaceApp and TikTok. *International Journal of Computer Science and Security (IJCSS)*, 14(2):38, 2020.

[21] R. P. Nogueira, J. C. Francisco, P. Domingues, and M. Frade. LabCIF/FAMA: Forensic Analysis for Mobile Apps, June 2020. https://doi.org/10.5281/zenodo.3906669 [Online; accessed on September 15, 2021].

[22] T. Pandela and I. Riadi. Browser forensics on web-based tiktok applications. *International Journal of Computer Applications*, 175(34):47–52, December 2020.

[23] K. Poulsen and R. McMillan. TikTok tracked user data using tactic banned by Google, August 2020. https://www.wsj.com/articles/tiktok-tracked-user-data-using-tactic-banned-by-google-11597176738 [Online; accessed on September 15, 2021].

[24] J. W. Rettberg. Hand Signs for Lip-syncing: The Emergence of a Gestural Language on Musical.ly as a Video-Based Equivalent to Emoji. *Social Media Society*, 3(4), Oct. 2017.

[25] B. Robert. TikTok: Logs, Logs, Logs, August 2020. https://medium.com/@fs0c131y/tiktok-logs-logs-logs-e93e8162647a [Online; accessed on September 15, 2021].

[26] B. Robert. TikTok: What is an app log?, August 2020. https://medium.com/@fs0c131y/tiktok-what-is-an-

app-log-da70193f875 [Online; accessed on September 15, 2021].

[27] F. Ryan, A. Fritz, and D. Impiombato. TikTok and WeChat - Curating and controlling global information flows. Technical Report 37, International Cyber Policy Centre / Australian Strategic Policy Institute, September 2020.

[28] S. Shead. Facebook owns the four most downloaded apps of the decade, December 2019. https://www.bbc.com/news/technology-50838013 [Online; accessed on September 15, 2021].

[29] A. Team. Autopsy 4, March 2021. https://github.com/sleuthkit/autopsy/blob/master/README.txt [Online; accessed on March 31, 2021].

[30] B. J. Tidy. Tiktok changes virtual gifts policy after bbc probe, Dec 2019. https://www.bbc.com/news/technology-50651125 [Online; accessed on September 15, 2021].

[31] J. Tidy. TikTok: What is the app and how much data does it collect?, August 2020. https://www.bbc.com/news/technology-53476117 [Online; accessed on September 15, 2021].

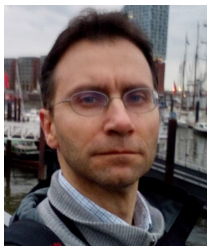
[32] TikTok. Strengthening privacy and safety for youth on TikTok, January 2021. https://newsroom.tiktok.com/en-us/strengthening-privacy-and-safety-for-youth [Online; accessed on September 15, 2021].

[33] D. Q. Tuan, S. Cheon, and Y. Won. On the io characteristics of the sqlite transactions. In *Proc. of the 2016 International Conference on Mobile Software Engineering and Systems (MOBILESoft'16), Austin, Texas*, page 214–224, New York, NY, USA, May 2016. ACM.

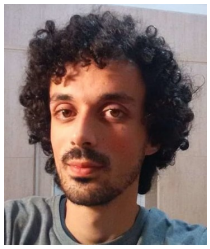
[34] N. Wang. *Data story of Tiktok*. Phd thesis, ARC III - Scuola del Design, Politecnico di Millano, http://hdl.handle.net/10589/152682 [Online; accessed on September 15, 2021], December 2019.

[35] G. Weimann and N. Masri. Research note: Spreading hate on TikTok. *Studies in Conflict & Terrorism*, pages 1–14, Jun 2020.

Authors Biography



Patricio Domingues holds a PhD (2009) in Informatics Engineering from the University of Coimbra, Portugal. He is currently an associate professor at the School of Technology and Management of the Polytechnic of Leiria, Portugal, where he teaches in the BSc in Informatics Engineering and in the Master of Cybersecurity and Digital Forensics. His main research interests include digital forensics, high performance, and many-core computing.



Ruben Nogueira is currently completing his BSc degree at the School of Management of the Polytechnic of Leiria, Portugal. He is currently working as a Full-Stack Web Developer and his main research interests include digital forensics, pentesting and cybersecurity.

José Carlos Francisco is currently a student in the Master in Cybersecurity and Digital Forensics at the School of Management of the Polytechnic of Leiria, Portugal. He holds a Bsc (2020) in Informatics Engineering also from Polytechnic of Leiria. His main research interests include digital forensics, pentesting and cybersecurity.

Miguel Frade holds a PhD (2012) in Informatics Engineering from the University of Extremadura, Spain. He is currently an adjunct professor at the School of Technology and Management of the Polytechnic of Leiria, Portugal, where he teaches in the BSc in Informatics Engineering, and in the Master of Cybersecurity and Digital Forensics. His main research interests include digital forensics, cybersecurity, and cryptography.