Forensic Science International: Digital Investigation 44 (2023) 301485

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

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

A forensic analysis of streaming platforms on Android OS

Julián García Murias, Douglas Levick, Sean McKeown*

School of Computing, Engineering & the Built Environment, Edinburgh Napier University, Edinburgh, UK

ARTICLE INFO

Article history:
Received 8 July 2022
Received in revised form
16 November 2022
Accepted 21 November 2022
Available online xxx

Keywords:
Streamed video forensics
Android application forensics
Cached video forensics

ABSTRACT

This work builds on existing research in streamed video reconstruction on the Android OS, which previously demonstrated that caching occurs in most cases for the Chrome and Firefox Web browsers. Prior work also outlined that streaming application caching behaviour is dependent on both the implementation of the service, as well as the actions taken by the user, with contrasting results between replaying videos and viewing live content. We conduct a forensic investigation for the Twitch, Facebook, Reddit, Instagram and Periscope Android applications, with a focus on the application specific folders in the/data/data directory. Applications were populated with data by creating accounts and viewing a mixture of live and replay (recorded) video streams, with a focus on attempting to recover video fragments or identifiers for particular streams/videos. As users may take action to hinder forensic endeavours, additional videos were viewed to identify baseline caching and overwriting behaviour on each application. Additionally, Android's 'Cache clear' operation was evaluated for its anti-forensic potential.

While Android seems to produce different behaviour for live and recorded streams, which is consistent with prior work, our findings suggest that Android applications typically retain few, or no, video artefacts, which contrasts with their browser based counterparts. Cache clearing also appears to be a powerful, and trivial, anti-forensics step for clearing locally cached media in each application. We suggest that, going forward, new applications should be tested on a variety of platforms, as it appears that they do not necessarily leave behind consistent forensic traces across versions.

1. Introduction

The Internet and social networks have modified the Digital Forensics landscape significantly, making investigations quite challenging due to the heterogeneous nature of social media (Arshad et al., 2019). Thus, images, videos and user interactions are potentially presentable as evidence in court. Moreover, the increasing usage of live streaming and its consequent abuse has emphasised the importance of forensic investigations in scenarios of user engagement with streamed media.

The popularity of personalised live streaming services has dramatically increased in the last decade. Their increasing usage has resulted in a number of recorded abuses, including harassment, offences against the person and the viewing/production of Child Sexual Abuse Material (CSAM), contrasting with the traditionally prevalent multimedia crimes pertaining to copyrighted materials (Horsman, 2019). In particular, the Internet Watch Foundation reported that 72% of actioned websites in 2021 (The Internet Watch Foundation, 2021) contained self-generated CSAM content, often originating from live-streaming services. The weaponisation of streaming services to communicate acts of extreme violence or terror has also been noted as an increasing trend (Conway and Dillon, 2016; Tikka et al., 2020). This has emphasised the importance of live streaming regulation and therefore, its forensic investigation. When conducting a live streaming forensic analysis, investigators must be aware of the current legislative developments, ensuring the type of information recovered will support criminal justice processes, as the digital forensics field is moving towards the utilisation of triage strategies to decrease the amount of time taken in carrying out an investigation (Horsman, 2016).

Research has been done in reconstructing streamed video content from the Web browser cache, showing that where a Web browser has been used to play a streamed video, stream content can be cached to a local device, and buffered video stream data can be reassembled to produce a viewable video clip of content (Horsman, 2018b). However, caching does not occur in every case, depending on the how streaming services work and on whether the

* Corresponding author.
E-mail address: s.mckeown@napier.ac.uk (S. McKeown).

https://doi.org/10.1016/j.fsidi.2022.301485

stream is a live stream or a replayed video (Horsman, 2018a).

The main aim of this study is to explore fundamental forensic questions surrounding the recoverability of cached content and metadata for streaming services on the Android OS, and how this is different from accessing the same platforms via a Web browser. The formulation of these questions (listed below) allows us to focus on practical outcomes for examiners, the answers to which we return to at the end of the paper (Section 6.1).

Forensic Questions:

1. **What are the forensic artefacts of user engagement for a variety of streaming services/applications on the Android platform?**
2. **How different are streaming platforms forensic artefacts in Android in relation to their Web browser versions?**
3. **Can streamed video content in Android OS be recovered and viewed?**
4. **Is it possible to determine how much of a video has been viewed in Android?**
5. **Can a user trivially adversely affect recoverability of locally stored artefacts?**

Thus, several streaming platforms (detailed in Section 3.2) were examined to determine the extent of latent evidence recovery in a scenario of user engagement with streamed video content. We also test the persistence of such artefacts after continued usage, and recoverability after a cache clear operation.

2. Related work

A live-stream is the audio and video transmission of an event over the Internet in real-time. Thus, live-streams need a medium that records and broadcasts in real time and a communication technology that allows images and sound to be sent instantly between two different locations (Chen and Lin, 2018). Due to the interactive nature of live-streams, streamers and viewers can interact, allowing communities to grow around a person or topic.

There is an existing body of literature in the field of reconstructing streamed video content using local cache sources, largely being conducted by Horsman (2018a,b, 2019). Horsman (2018b) studies the Web browser cache, demonstrating that where Chrome Web browser has been used to play a YouTube or Facebook Live video, video stream content can be cached to a local device, and buffered video stream data can be reassembled to produce a viewable video clip of content.

Building on the work on Chrome, and extending it to Firefox, Horsman (2019) analysed the Twitch, YouTube Live, Mixer, Ustream. tv, Smashcast. tv and Younow services, revealing that whilst caching occurs in most cases when playing a live stream, it does not happen for Younow, Facebook Live and Twitch. However, caching occurs in Twitch and Facebook Live when playing a stream replay. The above work emphasises the need to examine cached video content in those cases where Internet history indicates any sort of user engaging with streaming platforms.

Another case study (Horsman, 2018a) has been done in Periscope for the iOS, Android, and Web browser versions. Whilst the Web browser cache allows stream recovery, mobile application versions provide some investigatory challenges. In Android, where accessibility is greater, user activity is available, and cached content might be sufficient to attribute liability for an offence of the making or possession of indecent content. Despite this, streamed video cannot be reconstructed from Android's local cache. Artefacts related to a user's Periscope account can be found, such as username; account creation date and email; first and last names; number of broadcasts and viewed videos; and number of followers. Still images of broadcasts in . jpg format can also be found, potentially allowing the examiner to put them altogether to build a video recreation. In contrast, limited evidence was found in the corresponding iOS applications.

However, some limitations are present when it comes to creating a model or automating the reconstruction of streamed video content to produce a viewable video clip. According to the results of the YouTube Live analysis in Horsman (2018b), stream reconstruction is limited not to the content the user has viewed but the content which has actually been buffered. Therefore, if a user watches a 50s video, the last 10s cannot be reconstructed, as the content has not been buffered locally. Some traditional 'single-file' media analysis strategies are ineffective, as the individual stream fragment files are inconclusive when analysed alone. In some cases, this stream chunks are not even viewable. This is the case of Ustream. tv, where stream chunks are padded with data which prevent them from being played, and restoring chunks' original header is needed to reproduce them (Horsman, 2019). Thus, relying on stream artefacts associated metadata and specific stream attributes is needed (Horsman, 2018b). However, these cached fragments are different for each streaming platform, and determining a unique method is not possible. Additionally, local caching is only produced when the user is viewing stream replays for some streaming platforms, but not for those users viewing live broadcasts. This is the case for Facebook Live and Twitch. Finally, mobile device analysis present several challenges, as it has been outlined in the Periscope case study (Horsman, 2018a). Unlike iOS, Android allows to recover valuable information about streams, but cached content consists of still images rather than stream chunks. Indeed, for a comprehensive investigation of Periscope, live access to the application might be required, as comment based interactions are only accessible when the accessing the application live.

Finally, a study by AlZahrani et al. (2021) explores the local storage for the Android Twitch application in detail. From the SQLite database, XML and cache files, the username, email, phone number and IP of the user was recovered along with details of viewed streams, followed channels, and sharing interactions, together with timestamps. The authors were also able to identify recovery parsing methods via anchors (fixed-text strings). Our work demonstrates similar results for Twitch, which were included for the sake of providing a broad overview of the streaming application landscape, as well as to corroborate these existing findings.

2.1. Contextual considerations

Broadband connections have made streaming abuses unavoidable. In sections 48-50 of Sexual Offences Act (2003) (SOA), it is considered that a person (B) has been sexually exploited if an indecent image of B is recorded (or streamed or otherwise transmitted). Usually, streams which are hosted by a streaming service remain visible to other users, who could incur liability for sexual exploitation via streaming. On the contrary, the stream may be accessed by third parties (Person C) passively or in the form of recorded media once the live stream has finished. Despite no sexual exploitation occurring with respect to the specific wording in the SOA, Person C may still incur liability for an act of making or possessing indecent child imagery, according to the Protection of Children Act (1978) and Criminal Justice Act 1988 (Horsman, 2018a).

There are also some limitations when it comes to identifying potential sexual exploitation or other offences. In these cases, evidence of interaction between those involved in the offence is needed, as well as some sort of coercion. As outlined by Horsman (2018a) for Periscope, live access may be sufficient to get enough

evidence, but it probably would require the cooperation of the different parts involved and the access to both their accounts and their devices.

In general terms, prior work outlines that, contrary to the desktop streaming platform version, Android streaming applications provide attackers mobility, potentially increasing the number of networks the attacker is able to penetrate and thus, the number of exposed users. Additionally, the usage of these widespread streaming services can lead to a variety of security and privacy implications (Nikas et al., 2018), meaning that such services, and their applications, should be examined carefully to best facilitate forensic investigations.

3. Methodology

Existing work in streamed content reconstruction leaves a promising landscape, but lacks research in streaming mobile applications to support the findings of browser-based analyses. The aim of this study is to determine how much evidence can be obtained in a forensic investigation related to user engagement with mobile streaming platforms, and how different are these Android forensic artefacts in relation to their Web browser versions, while also considering cache-clearing as an anti-forensics technique. To facilitate these endeavours, a selection of popular, relevant, Android streaming applications were chosen for in-depth analysis. As the study involves the recovery of potentially deleted data it was necessary to acquire bit-level, physical, copies of real device storage, as opposed to virtualising the experiments.

3.1. Mobile device configuration and acquisition

The physical device, which was populated with data and forensically acquired, was a BQ Aquaris X Pro, released in2017.[1] BQ was a Spanish brand, which ceased trading in 2021, whose Android ROMs are very close to Android stock, contain Google Services, and have little customisation. This relatively 'vanilla' build means that it should be a representative candidate for the wider Android ecosystem.

Performing a complete forensic acquisition of a mobile device is a challenging process, as each smartphone model, manufacturer and investigation scenario is unique. As physical level device access is required the device needs to be rooted. Rooting enables users to perform high privileged functions which are not allowed for ordinary users (Grover, 2013), such as accessing the underlying storage device directly. This was achieved using *Magisk v20.4*.[2] In order to exploit access, a custom recovery partition must be flashed on the device. TWRP Recovery has a specific release for BQ Aquaris X Pro device, 3.3.1–0 (bardock).[3] Android device and rooting software details are provided in Table 1. Before the whole process starts, Google and BQ drivers must be installed in the forensic workstation. With the elevated privileges of the root user, the device was forensically acquired using *Magnet AXIOM*. Autopsy and its Android Analyzer Ingest Module were used in subsequent processing and analysis.

3.2. Application choice

Some of the most widely used streaming services have been selected to be analysed: Facebook Live, Reddit streaming, Instagram live, Twitch, YouTube live and Periscope. Streaming services have been chosen taking into account the number of users, impact on the Internet, and the different streaming emphases (such as in the case of Reddit and Instagram Live, which are not de facto streaming platforms, but provide live streaming functionality). The streaming applications which were previously analysed in the literature have been included to allow for direct comparisons. A brief description of the chosen platforms is provided below for context, with application version numbers being provided in Table 2.

Table 1
Android device, rooting and forensic processing software version information.

Software	Version
Android Device	
Android version	8.1.0
Build number	2.11.0_20191121-1510
Kernel version	3.18.71-perf-g4870138 (gcc v4.9 xx20150123)
Device Rooting	
Rooting software	Magisk v20.4
Recovery software	TWRP Recovery 3.3.1–0
Forensic Tools	
Autopsy	4.15.0
Magnet AXIOM	4.1.1.20153

Table 2
Application version and release date for all streaming applications tested.

Streaming Platform	Version	Release Date
Twitch	9.2.0 (902000)	June 11, 2020
YouTube Live	15.22.35 (1512955328)	June 8, 2020
Instagram Live	144.0.0.25.119 (217948947)	June 2, 2020
Facebook Live	273.0.0.39.123 (218047935)	June 4, 2020
Reddit	20.20.21.1	June 10, 2020
Periscope	1.30.00 (2100520)	June 4, 2020

- **Twitch**: Introduced in 2011 as a spin-off of Justin TV, Twitch is a video streaming service which operates as a subsidiary of Amazon. It is considered as the leading live streaming video service for video games (140 million active users (Brian Dean, 2021)), especially in the US. In addition to its iOS and Android apps, Twitch provides a Desktop version for MacOS and Windows.
- **YouTube Live**: YouTube is one of the most powerful and popular video hosting services (2.5 billion active users (DataReportal, 2022)), especially since November 2006, when he was acquired by Google and started to operate as one oCheeky4n6monkey, 2014f its subsidiaries. YouTube allows its users to upload, view, rate, add to playlists, report, comment on videos and subscribe to other user's channels.
- **Instagram Live**: Now owned by Facebook, Instagram is one of the most popular photo and video-sharing social networking services all over the world (approximately 1.5 billion active users (DataReportal, 2022)). In 2018, Instagram launched IGTV as a vertical standalone video application for smartphones. IGTV allows users to load videos up to 10 min/650 MB or 60 min/3.6 GB for verified and popular accounts.
- **Facebook Live**: Facebook started to allow users to live stream video in 2015. Since then, it has been a direct competitor to services such as Periscope. Users are allowed to engage with streams and through the News Feed and comment on them in real-time. Facebook itself has approximately 3 billion active users (DataReportal, 2022).
- **Reddit**: Founded in 2005, Reddit is an American social news aggregation, Web content rating and discussion Website. Users usually submit content such as links, text posts and images, which are rated by the rest of users. Reddit is structured in "subreddits". Recently, Reddit announced the

[1] https://www.gsmarena.com/bq_aquaris_x_pro-8641.php.
[2] https://magisks.com.
[3] https://twrp.me/bq/bqaquarisxpro.html.

"Reddit Public Access Network", which lets users live-stream to a new subreddit, with popular streams appearing on the front page. Reddit has around 430 million active users (DataReportal, 2022).

- **Periscope**: Acquired by Twitter before launch in 2015, Periscope was previously one of the most popular live-streaming applications. It no longer appears on top usage statistics, but is included here for breadth. Users can select to share their videos publicly or make them available for just a few users.

3.3. Scenario setup and forensic process

The following steps were taken to populate and acquire data, **with a forensic image being captured at each stage**:

1. Each application was installed on the device and a clean, **baseline**, image was acquired. This allows for pre-existing artefacts to be discounted in the analysis.
2. **Live streams and replays were viewed** in each application. Streams are accessed in their standard way, no other playback options have been explored. Each video was viewed for a median of 33 s, regardless of whether the stream was live or a replay.
3. As an anti-forensics technique, the applications were used heavily to view many streams, following links, threads, and generally browsing. This subjects the **application cache** to **heavy usage**, and potentially churn, allowing us to assess the impact of high usage on data recovery.
4. Lastly, in a direct attempt to purge data directly, the application **cache was cleared** (via the standard Android application management interface).

Together, these four images allow us to assess evidence retention in both normal use cases, as well as a deliberate attempt to purge application specific evidence.

4. Findings

4.1. Investigation of forensic artefacts

The *Autopsy* tool was used to analyse the raw image generated by *Magnet AXIOM* in search for relevant forensic artefacts. At the time of writing, *Autopsy* fully supports both the EXT4 filesystem used by the device, and some Android artefacts (via an ingest module) directly. Each streaming application directory, located in the `/data` partition, was analysed, as well as a datetime-range search to identify any additionally relevant artefacts on the filesystem.

What follows is a detailed analysis of the findings for each application in turn, an overview of which is provided in Section 4.2.

4.1.1. Twitch

For *Twitch*, forensic artefacts are found in `/data/tv.twitch.android.app` folder. Several artefacts can be found, beginning with valuable data extracted from the `/shared_prefs` sub-directory, which is used to store key–value pairs in Android:

Listing 1: Example contents of the Twitch file/shared_prefs/recent_searches.xml containing recent stream searches.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="recentSearches">[&amp;quot;Fortnite&amp;quot;]</string>
</map>
```

Listing 2: Example contents of the Twitch file/shared_prefs/banner.xml which tracks stream watch time.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<int name="minutes_watched" value="2" />
</map>
```

Listing 3: Example contents of the Twitch file/shared_prefs/recently_watched.xml for recently viewed items.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
        <long name="last_watched_game_id" value="33214" />
        <string name="recentlyWatchedGames">
        {&amp;quot;32982&amp;quot;:1591967604026,
        &amp;quot;33214&amp;quot;:1592932837569}</string>
</map>
```

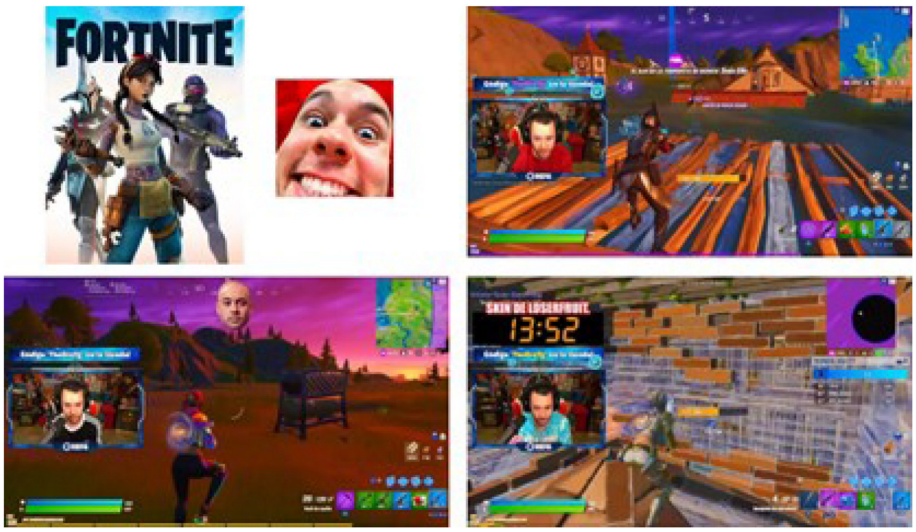

Fig. 1. Twitch cached images for viewed a category (top-left) and stream replays.

The *recent_searches.xml* (Listing 1) file contains reference to previous search terms, with *banner. xml* (Listing 2) providing a record of the time spent watching videos. *recently_watched.xml* (Listing 3) specifies pairs of `game_id` and the timestamp of the last time a stream from that game was viewed. `game_id` can be found using the Twitch API[4] but we also discovered a Github page (Nerothos, 2022) which has (at the time of writing) an up-to-date mapping of `game_id` to human readable names. From the experiment we were able to verify that 32982 for GTA V and 33214 for Fortnite is accurately obtainable from the list.

Some images were found to be cached in `/cache/image_manager_disk_cache`. The images (see examples in Fig. 1) correspond to every Twitch stream which is listed, and when a search is performed, the stream categories which are listed and the profile pictures of each channel. Then, images from the replayed streams are cached. For a video replay, several images corresponding to the viewed stream were cached. A preview is cached for the rest of the listed streams. In order to establish the time in which streams have been viewed, creation, modification and last accessed timestamps of cached images need to be examined.

4.1.2. YouTube Live

YouTube artefacts are located in `/data/com.google.android.youtube`. Two main sources of data are found: `/cache` and `/files`. In `/files` several items provide some valuable information related to the user account and the videos usage. One file, `MANIFEST-000002`, was recovered from unallocated space.

Listing 4: Example contents of YouTube's/files/watch/shared/MANIFEST-000002.

```
+34XXX XXX XXX [redacted]
TY24 +34XXX XXX XXX [redacted]
user@gmail.com [redacted]
```

Listing 5: Example contents of YouTube's/files/offline.

```
https://www.pubg.com/privacy/! for PUBG.
You're connecting as [redacted User name]
:. You can disconnect at any time in your YouTube Settings.
pubg.com
```

Listing 6: Example contents of YouTube's/shared_prefs/com.google.android.gms.measurement.prefs.xml. The timestamp is

[4] From the https://api.twitch.tv/helix/games endpoint, specifically.

stored in Unix Epoch format.

```
<long name="last_pause_time" value="1592933514046" />
```

Listing 7: Example contents of YouTube's/files/media/shared/media.pb. We were unable to determine the timestamp format.

```
last_playback_start_timestamp
C890K9gBZPF_QBNw24
```

Listing 8: Example contents of YouTube's/files/player/shared/playability_settings.pb. We were unable to determine the timestamp format.

```
@playability_adult_confirmation_time_stamp:
cM-IHPLfll6wj3WLXrtlQ
```

The deleted *MANIFEST* (Listing 4) file contains the user's mobile phone numbers and email address (redacted above), while the name of the connected user is available in*/files/offline* (Listing 5), together with some contextual information for when it was displayed. Several timestamps were recovered. The *GMS prefs. xml* (Listing 6) contains a Google Mobile Service (GMS) timestamp, which appears to be for the purposes of analytics, stored in integer format as an epoch time. *media. pb* (Listing 7) contains the last playback timestamp, while *playability_settings.pb* (Listing 8) records a timestamp for age verification confirmation. We were unable to identify the formatting, though it does appear to be similar to Google's ei parameter encoding (Cheeky4n6monkey, 2014).

The largest source of evidence for the *YouTube* application is the `/cache` folder. Within it, cached content is split in to multiple `.cache` files. These files cannot be opened with a regular Windows/Linux application, but they can be carved to obtain images, as YouTube stores images as serialised Java Objects. By opening these files in binary, they can be explored until a known file signature is found. We identified three types of file signatures in the cache:

JFIF: FF D8 FF E0 - `JPEG/JFIF` graphics files.

RIFF: 52 49 46 46 - `WebP` Google WebP image files.
PNG: 89 50 4E 47 0D 0A 1A 0A - `PNG` Portable Network Graphics files.
HDR: File headers, only contains text.

An example cache file is shown in Fig. 2. Carved image files were viewable in supported applications (a Web browser was used for WebP), while the HDR files were viewed in a text editor.

Finally, a file named `zeroprefixparsed.cache` (Listing 9) stores the search history, as seen in Listing 9:

Listing 9: Example contents of the file zeroprefixparsed. cache containing the YouTube search history.

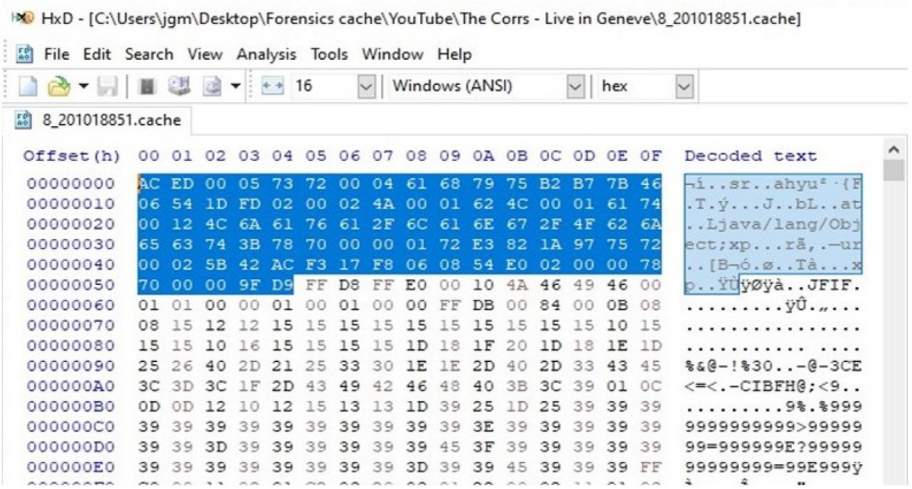

Fig. 2. YouTube Live cache file. A JFIF header can be seen immediately after the highlighted area.

Fig. 3. Image corresponding to a Thumbnail of the viewed *YouTube* replayed video.

```
<b>the corrs live in geneva</b>
w/complete/deleteitems?client=youtube-android-pb&delq=the+cor
↪  rs+live+in+geneva&deltok=AKtL3uQGoGWIrV78OSnp2fJF9deHlFdF
↪  JQ:
```

For live videos, we recovered a cached item for a stream preview (as it appears in the stream list, e.g., after a search). Correlated with this stream were images for user profiles (including the test account) which have interacted with the channel and stream.

More evidence appears to be available from stream replays, where several video screenshots of stream content were recovered. Each image contains 25 small previews in a 5 × 5 grid (e.g., Fig. 3). These images appear to be the timeline previews a user would see when hovering a cursor/finger over the video progress bar.

4.1.3. Instagram Live

Instagram artefacts were recovered from `/data/com.instagram.android`. Firstly, several items were recovered from `/app_state_logs`:

- Log files `.v9.txt_entity` and `.v9.txt_wrotedump`, were recovered from unallocated space. Despite the names, both files consisted of the same image, showing evidence of a live stream being available on the stories feed to be viewed.
- Log files `.v9.txt` and `.v9.txt_anr` both contain logs of the Instagram app activities, such as "explore", "searches" and "subscriptions", each being accompanied by a timestamp. The very last logged activity has a "last_active_time" timestamp which can be converted to human-readable date.

Instagram caches forensically valuable data in its `/cache` folder. Two main sources of information are available: `/images` and `/ExoPlayerCacheDir`. In the `/images` folder `.clean` and `.tmp` files are stored, which include still images of user profile pictures, publications, stories, advertisements and video screenshots. `/ExoPlayerCacheDir` contains chunks of cached content, including Instagram stories, advertisements, live streams and replayed videos. These video fragments are stored in . exo format, and largely appear to be encrypted and compressed[5] in real-time when they are broken into parts, possibly as a copyright control mechanism. However, some of these `.exo` chunks contain unencrypted data and are actually playable. By changing every single chunk to `.mp4` extension, some of them are viewable with a regular media player such as Video Lan Media Player. For the experimental scenario, three of these playable chunks corresponded to audio files of approximately 3 s in length depicting audio segments from viewed streams (both live and replays).

Finally, some other valuable data is cached, which is not detailed here, but pertains to: pending comments, pending follows, pending likes, pending assets, JSON files containing raw response data, etc.

4.1.4. Facebook Live

Facebook Live evidence is located in `/data/com.facebook.katana`. No relevant evidence was found for live streams. In contrast, videos from the user's newsfeed and replayed videos are cached in `/Data/ExoPlayerCacheDir/videocache`, though we did not find a way to correlate them with the user who posted

[5] Much in the same way that Google appears to do for their YouTube application (Fisher, 2022).

them. Similarly to Instagram, these videos are cached in `.exo` chunks, and some of them are viewable. The difference is that some of them are viewable as still images by Autopsy. Again, bearing similarity to Instagram, some other evidence is available, such as pending follows, pending likes, pending comments, etc.

Finally, since *Facebook* uses MQTT (MQ Telemetry Transport) to enable messages to be delivered efficiently in milliseconds, logs about timeouts, connection retries, dns lookup durations or MQTT version info can be found in `/cache/mqttlog_eventn.txt` (with n being an integer counting from 0, see Listing 10), which could potentially offer information related to the wireless network used as a connection source. This might provide information about the user location at a specific moment. An example of a Facebook MQTT logged event is provided in Listing 10.

Listing 10: A Facebook network exception event in mqttlog_eventn.txt.

```
06-23 00:17:54.521 [mqtt_disconnection_on_failure]
↪ exception=TimeoutException; reason=OPERATION_TIMEOUT;
↪ network_extra_info="WLAN XX09"; is_in_idle_mode=false;
↪ network_state=CONNECTED; network_subtype=;
↪ network_type=WIFI; operation=PING;
```

4.1.5. Reddit streaming

Reddit evidence is found in `/data/com.reddit.frontpage`. Two relevant directories contain valuable data: `/image_manager_disk_cache` and `/subreddit_listing`.

In `/image_manager_disk_cache`, several images are cached, corresponding to user profile pictures, subreddit images, posts, Reddit logos or adverts. If a video is played, a screenshot might be cached by the Reddit app.

`/subreddit_listing` contains relevant information about the nature of the stream interactions in files such as:

`SubredditListingKey(path=f.a.h.g.e@2bcb021)` (Listing 11).

Listing 11: Example contents of the SubredditListingKey(path = f.a.h.g.e@2bcb021) file. Facebook group ID redacted.

```
[{"id":"2t6pv","name":"t5_2t6pv",
↪ "display_name":"YouTubeGamers",
↪ "display_name_prefixed":"r/YouTubeGamers", "icon_img":"",
↪ "key_color":"", "banner_img":"",
↪ "header_img":"https://a.thumbs.redditmedia.com/
V8NWVx5TNAtqfl7f.png", "title":"Youtube Gamers - Reddit
↪ Youtube Gaming Community", "description":"r/YouTubeGamers
↪ is for YouTubers looking to show off their videos  to
↪ gain subscribers and increase views! Self-promotion is
↪ allowed! You can also just share other gaming videos you
↪ think are awesome!\n\nWe also have a Facebook community
↪ at: https://www.facebook.com/groups/<redacted>/\n\n.}]
```

However, not all subreddits which were tested resulted in residual evidence traces being retained.

4.1.6. Periscope

Periscope evidence is found in `/data/tv.periscope.android` folder. Cached data is stored in `/cache/image_manager_disk_cache`. Some images are cached, corresponding to stream previews when a set of streams is listed. These images are cropped, as most streams are displayed in 16:9 size, and do not fit in the preview thumbnail. Also, the same image is cached if the stream is viewed. Stream preview thumbnails are constantly updated (according to the stream progress), and all the subsequent thumbnails are also cached. Moreover, channel profiles are also cached.

4.2. Application summary and discussion

Results have shown that despite some sort of caching occurring in almost every case, it is not always possible to determine whether those cache artefacts correspond to a viewed stream or not. Table 3 shows a summary of the forensic artefacts created by the evaluated Android applications. Roughly speaking, it can be stated that live streamed videos are not cached. Instagram is the only exception, though it seems to be related to the way the application operates, as it caches everything regardless of whether it is a publication, story or a live video. When it comes to stream replays, caching usually occurs. However, caching does not happen for Twitch and Reddit. This is not a surprise in the case of Reddit, as it does not provide a 'live stream replay' feature, but it is unexpected in the case of Twitch, as caching occurs for its desktop Web browser version for both Firefox and Chrome (Horsman, 2018b, 2019).

Live streaming services in Android appear to make it difficult for

Table 3
Forensic artefacts summary for all streaming applications tested.

Streaming Platform	Video caching?	Cached Stream Artefacts	Possible to determine if a stream has been viewed?
Twitch	No.	Recent searches, minutes watched, recently watched games ids, screenshots of categories, channel profile pictures and stream previews.	No, only which streams and categories have been listed.
YouTube Live	Only for video replays	Last pause time, .cache files corresponding to PNG, JPEG/JFIF and RIFF/WebP files, search history in zeroprefixparsed.cache file.	No, it can only be determined how much of a video replay has been buffered.
Instagram Live	Yes.	Still images of profile pictures, publications and stories, screenshots of live streams and user newsfeed (/files directory). Encrypted non-viewable .exo files corresponding to video and user stories caching (some of them partially viewable).	No.
Facebook Live	Only for stream replays.	.exo files corresponding to the user newsfeed and replayed videos (some of them viewable as images with Autopsy.)	No.
Reddit	No.	Subreddits listing, still images of user profile/subreddit, logos, images, posts and streams/videos screenshots.	No.
Periscope	Only for stream replays.	Still images containing video fragments for and video previews.	No, it can be only determined how much of a video replay has been buffered.

a forensic examiner to determine whether a stream has been viewed or not. At most, it can be determined how much of a stream has been buffered for stream replays on the Periscope and YouTube Live platforms. The same could be possible for Instagram Live and Facebook Live if there is a method to decrypt and play cached . exo files. As YouTube's offline saved videos are also stored in . exo encrypted chunks, the same approach appears to be used in these cases. Since video buffering can only occur if the user accessed the video, evidence of video buffering might be sufficient to attribute liability in a scenario where there is suspicion of user engagement with illegal or indecent streamed media. However, a user can access a stream and pausing it without actually viewing it.

Moreover, despite recovered streaming artefacts being different across each platform, some indicators of user activity are common. In general, user profile pictures are cached locally, as well as images related to the viewed streams. Thus, cached elements having a specific modification or access date timestamp might suggest user activity for that app or streaming service at that specific moment. However, as most apps are able to run in the background and cache elements automatically, some noise will unequivocally be generated from the forensic point of view. In order to support evidence, "recent searches" or other user activity data might be necessary.

5. Anti-forensics of cached data

As cached images and video clips seem to be particularly important and consistent sources of evidence in Android streaming applications, it was prudent to investigate artefact retention in scenarios where the user has taken action to hinder recoverability. Two approaches, both easily accessible to Android users, were explored: *i)* additional application usage to fill and overwrite the cache; and *ii)* performing an application cache clear operation.

5.1. Continued use and additional caching

By continuing to use the application some artefacts might be overwritten, while also introducing noise to increase the difficulty of correlating artefacts with particular streams. Further evidence generation consisted of viewing over an hour of video from multiple sources on each streaming platform.

Little effect was observed for most of the tested applications. For those apps which are not caching content, such as Twitch, Reddit or

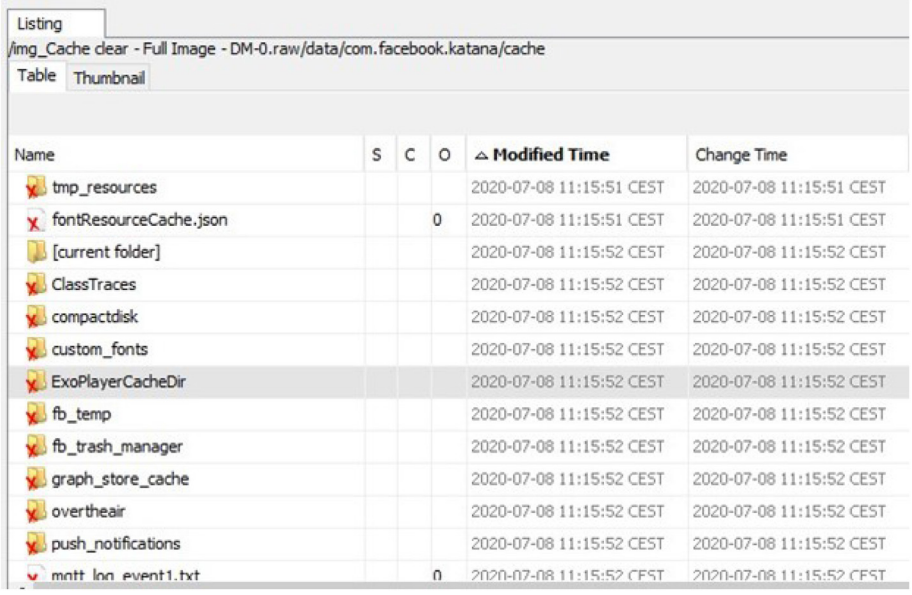

Fig. 4. Facebook`/cache` directory viewed in Autopsy after a cache clear operation.

Facebook Live (for live videos), there was no difference in terms of artefacts recoverability. For platforms which cached replayed videos, such as Periscope, Facebook or Instagram Live, additional items were populated into the cache without overwriting existing data. After this test, Instagram had stored 260 MB of cache and Facebook 93.10 MB, Periscope stored only 17.58 MB (only still images are cached), and Twitch 2.61 MB (only a few screenshots were cached).

Despite some work studying the structure of Android caches (Immanuel et al., 2015; Kim et al., 2019) we were unable to determine if there is a maximum size for Android application caches which would cause elements to be removed. Anecdotally, application caches can reach hundreds of megabytes, as is reflected by many consumer articles online pertaining to cache clearing as a means of freeing up device storage space (for example: Domenic Molinaro (2021); Szewczyk and Haskell-Dowland (2021)).

5.2. Cache clearing

The application specific cache was cleared for all services, which can be achieved via a long press on the application and selecting App Info → Storage → Clear cache, or by accessing the same application specific menu via Settings → Apps or the Storage menu.

After caches were cleared, a full image of the Android device was acquired. While Autopsy was able to recover deleted directories in the application caches, files are largely absent. An example cleared cache directory for Facebook is shown in Fig. 4. In cases where files were still recoverable, they contained no content. An `istat` command executed for one of the empty files (previously a cached image) shows that the deletion timestamp corresponds to the cache clear event.

This demonstrates that cache clearing operations can be effective at purging residual evidence for media-heavy applications, particularly when the application itself does not keep verbose records of user activity in local storage. The TRIM operation is likely responsible for the non-recoverability of file data, even when a record is still recoverable from the filesystem metadata.

6. Evaluation of forensic questions and conclusions

In this section we will discuss the implications of our findings for relevant forensics questions (generalising across all tested streaming applications), before drawing general conclusions.

6.1. Forensic questions

1. **What are the forensic artefacts of user engagement for a variety of streaming services/applications on the Android platform?**
   Several artefacts can be found, depending on the application. In general, it is possible to recover still images relating to user profile pictures, stream categories or stream preview thumbnails.

   Stream or video search history is present for some applications, as well as a last paused timestamp (*YouTube*). For other platforms like *Reddit*, a subreddit listing is also stored. Finally, encrypted caching chunks in `.exo` format can be recovered for some applications (Facebook, Instagram). Most of these chunks are not viewable.

2. **How different are streaming platforms forensic artefacts in Android in relation to their Web browser versions?**
   Streaming platforms forensic artefacts in Android are completely different from their Web browser versions, as cached video chunks are not present, but still images, logs and preferences files. The only exception is *Facebook*, which stores encrypted `.exo` files in its caching folder.
3. **Can streamed video content in Android OS be recovered and viewed?**
   Generally, no. Most applications do not cache streamed videos in local application directories, and those which perform some sort of caching only allow to recover a few still images (*Periscope*, *YouTube*). However, stream reconstruction would be possible for some applications (*Facebook* or *Instagram*) if there was a way to decrypt and resemble together `.exo` chunks.
4. **Is it possible to determine how much of a video has been viewed in Android?**
   No. At most, it is possible to determine how much of a video has been buffered. However, this is only possible for video replays for the *YouTube* and *Periscope* applications.
5. **Can a user trivially adversely affect recoverability of locally stored artefacts?**
   Yes, the cache clearing operation appears to be effective in purging application data, making it seemingly unrecoverable in our tests. It is unclear how long cache data persists in normal operation, however.

6.2. Conclusions

Streamed artefacts have an enormous potential in forensic investigations in which there is suspicion of indecent content possession or engagement. However, recoverability has been found to be lower than expected based on prior work conducted on periscope (Horsman, 2018a). Moreover, large scale streamed media reconstruction appears to be infeasible across all tested applications, which appears in stark contrast to the browser-based versions of similar applications (Horsman, 2018b, 2019). It is therefore advisable that a cross-platform forensic study is conducted for new, or important, applications.

As media assets must be stored somewhere to be displayed, or to allow for the user to move around a video timeline, it appears likely that Android applications have a bias towards caching assets in memory (noted as an option in Kim et al. (2019)), rather than writing to disk. It should, however, be noted that the focus of our work was on the application local specific data stores (in `/data/data`), and it is possible that some items are present in caches for the Android Runtime (ART) environment, or other API caches (Immanuel et al., 2015).

When it comes to drawing conclusions about the possibility of stream viewing attribution, the biggest challenge is the fact that even though the evidence of streams viewing is usually recoverable, there is often no clear way to differentiate streams and videos which have been viewed, as opposed to simply listed or displayed in an overview pane. However, platforms such as YouTube and Periscope allow us to determine whether a stream replay has been buffered or not.

Further work would ideally explore popular streaming applications on iOS and other relevant platforms, as the behaviour we noted was surprising, and may indicate that platforms have trends and varied recoverability. Additionally, we note that Android is a popular option for home media centre devices and offers the capability to be remotely controlled, creating further problems of attribution (Morrison et al., 2017).

Finally, further work into automating the process of extracting artefacts, which are quite similar for each application, should be pursued. This may include reversing the serialisation of objects (such as that which occurs in YouTube streams), understanding the composition of . exo artefacts, or simply identifying regular

expressions which can be used to identify fixed-text signatures, similar to prior work on Twitch (AlZahrani et al., 2021).

Data availability

The data that has been used is confidential.

References

AlZahrani, A., Wani, M.A., Bhat, W.A., 2021. Forensic analysis of Twitch video streaming activities on Android. J. Forensic Sci. 66, 1721–1741. https://doi.org/10.1111/1556-4029.14750.

Cheeky4n6monkey, 2014. Cheeky4n6Monkey - learning about digital forensics: google-ei'd. URL: https://cheeky4n6monkey.blogspot.com/2014/10/google-eid.html.

Chen, C.C., Lin, Y.C., 2018. What drives live-stream usage intention? the perspectives of flow, entertainment, social interaction, and endorsement. Telematics Inf. 35, 293–303.

Conway, M., Dillon, J., 2016. Future trends: live-Streaming terrorist attacks VOX Pol. URL: https://www.voxpol.eu/download/vox-pol_publication/Live- streaming_FINAL.pdf.

DataReportal, 2022. Digital 2022 April Global Statshot Report. Apr 2022) v01. URL: https://www.slideshare.net/DataReportal/digital-2022-april-global-statshot-report-apr-2022-v01.

Dean, Brian, 2021. Twitch usage and growth statistics: how many people use twitch in 2022? URL: https://backlinko.com/twitch-users.

Fisher, T., 2022. What's an EXO file and how do you open one? URL: https://www.lifewire.com/exo-file-2621155. section: Lifewire.

Grover, J., 2013. Android forensics: automated data collection and reporting from a mobile device. Digit. Invest. 10, 12–20. https://doi.org/10.1016/j.diin.2013.06.002.

Horsman, G., 2016. Digital forensics: understanding the development of criminal law in england and wales on images depicting child sexual abuse. Comput. Law Secur. Rep. 28, 419–432. https://doi.org/10.1016/j.diin.2019.01.009.

Horsman, G., 2018a. A forensic examination of the technical and legal challenges surrounding the investigation of child abuse on live streaming platforms: a case study on periscope. J. Inf. Secur. Appl. 42, 107–117. https://doi.org/10.1016/j.jisa.2018.07.009.

Horsman, G., 2018b. Reconstructing streamed video content: a case study on youtube and facebook live stream content in the chrome web browser cache. Digit. Invest. 26, 30–37. https://doi.org/10.1016/j.diin.2018.04.017.

Horsman, G., 2019. Reconstructing cached video stream content:- part 2. Forensic Sci. Int.: Digit. Invest. 31. https://doi.org/10.1016/j.fsidi.2019.200893.

Immanuel, F., Martini, B., Choo, K.K.R., 2015. Android cache taxonomy and forensic process. In: 2015 IEEE Trustcom/BigDataSE/ISPA, pp. 1094–1101. https://doi.org/10.1109/Trustcom.2015.488.

Kim, H., Kim, D., Jo, W., Shon, T., 2019. Digital forensic analysis using android application cache data. In: 2019 International Conference on Platform Technology and Service. PlatCon), pp. 1–4. https://doi.org/10.1109/PlatCon.2019.8669409.

Molinaro, Domenic, 2021. How to clear your android cache & why you should do it. URL: https://www.avast.com/c-clear-cache-android.

Morrison, L., Read, H., Xynos, K., Sutherland, I., 2017. Forensic evaluation of an amazon fire tv stick. In: Advances in Digital Forensics XIII: 13th IFIP WG 11.9 International Conference, pp. 63–79. https://doi.org/10.1007/978-3-319-67208-3_4. Orlando, FL, USA.

Nerothos, 2022. TwithGameList. URL: https://github.com/Nerothos/TwithGameList. original-date: 2018-12-22T17:51:50Z.

Nikas, A., Alepis, E., Patsakis, C., 2018. I know what you streamed last night: on the security and privacy of streaming. Digit. Invest. 25, 78–89. https://doi.org/10.1016/j.diin.2018.03.004.

Szewczyk, P., Haskell-Dowland, P., 2021. What is 'Other' in my iPhone storage, why is it taking up so much space and how do I clear it? URL: http://theconversation.com/what-is-other-in-my-iphone-storage-why-is-it-taking-up-so-much-space-and-how-do-i-clear-it-160994.

The Internet Watch Foundation, 2021. IWF: the Annual Report 2021. URL: https://annualreport2021.iwf.org.uk/pdf/IWF-Annual-Report-2021.pdf.

Tikka, M.T., Sumiala, J., Harju, A., Valaskivi, K., 2020. Weaponization of Liveness: Streaming Death as a Hybrid Media Event of Terrorist Violence. AoIR Selected Papers of Internet Research.