

# Security, Privacy and Steganographic Analysis of FaceApp and TikTok

**Ashar Neyaz**

Department of Computer Science  
Sam Houston State University  
Huntsville, TX, USA

[ashar.neyaz@shsu.edu](mailto:ashar.neyaz@shsu.edu)

**Avinash Kumar**

Department of Computer Science  
Sam Houston State University  
Huntsville, TX, USA

[avinash@shsu.edu](mailto:avinash@shsu.edu)

**Sundar Krishnan**

Department of Computer Science  
Sam Houston State University  
Huntsville, TX, USA

[skrishnan@shsu.edu](mailto:skrishnan@shsu.edu)

**Jessica Placker**

Department of Computer Science  
Sam Houston State University  
Huntsville, TX, USA

[jdp085@shsu.edu](mailto:jdp085@shsu.edu)

**Qingzhong Liu**

Department of Computer Science  
Sam Houston State University  
Huntsville, TX, USA

[qx1005@shsu.edu](mailto:qx1005@shsu.edu)

---

## Abstract

Smartphone applications (Apps) can be addictive for users due to their uniqueness, ease-of-use, trendiness, and growing popularity. The addition of Artificial Intelligence (AI) into their functionality has rapidly gained popularity with smartphone users. Over the years, very few smartphone Apps have quickly gained immense popularity like FaceApp and TikTok. FaceApp boasts of using AI to transform photos of human faces using its powerful facial recognition capabilities. FaceApp has been the target of ensuing backlash against it driving the market for a number of other similar yet lesser-known clones into the top ranks of the App stores. TikTok offers video editing and sharing of short video clips whereby making them charming, funny, cringe-inducing, and addictive to the younger generation. FaceApp and TikTok have been the targets of the media, privacy watchdogs, and governments over worries of privacy, ethnicity filters, data misuse, anti-forensics, and security. In this paper, the authors forensically review FaceApp and TikTok Apps from the Android Play Store, for their data ownership, data management, privacy concerns, steganographic use, and overall security posture.

**Keywords:** Mobile Forensics, Smartphone Forensics, TikTok, FaceApp, Steganography, Privacy, Security, Cybersecurity.

---

## 1. INTRODUCTION

Global rise in mobile data use has had an impressive trajectory because of customer demands over the years. Cellular Telecommunications Industry Association's (CTIA) 2019 annual report [1] showcased the growth metrics from 2017 to 2018 with an unprecedented 82% increase. The report also shows that the number of smartphones in the U.S. being used went from 273.2M to

284.7M which is a 4% increase. The Pew Research Center for Internet and Technology reports that as of June 2019, 81% of all Americans owned a smartphone [2]. Each variety of cellphone requires a specific variety of evidence collection tools and techniques. Furthermore, they all have varying mobile apps that collect important data that can be used by forensic phone examiners. As the complexity in smartphones and their Apps grow, so must the tools utilized to extract evidential data from them.

Forensic investigations concerning someone's digital footprint are usually of a time-sensitive nature. Often, law enforcement agencies must use forensic tools in order to recover digital evidence from crimes committed. "Recently the U.S. Supreme Court ruled that a warrant is required for most searches of cell phones at a scene" [3]. At times investigative agencies must request the disclosure of personal information such as calls, texts, and GPS information from 3rd party organizations which sometimes requires lengthy court orders. Further challenges concerning the use of digital forensic tools are the chain of custody and the experience needed to navigate devices (evidence) without altering any information, which could then potentially render the evidence as inadmissible in criminal or civil legal court cases [3].

Electronic or Digital Forensics (E-forensics) can be defined as the process of identifying, observing, preserving, and analyzing digital evidences by complying with standard operation procedures [3]. On the other hand, Anti-forensics closely resemble data concealment by hiding evidences or messages into devices to thwart detection and imperceptibility [4]. One way for potential criminals to hide information or evidence is by using a technique called steganography. This practice conceals a secret information within something else. For example, hiding a secret file or message within a digital photo that renders the altered photo to look almost the same as the original. Although there are many flavors of digital forensic tools, there remain limitations on successfully detecting hidden data using steganographic methods. There are many tools freely available to assist newbies and malicious users with stenography techniques. Such tools can be employed to conceal information on multi-media data and transmitted via smartphones. Often text messages, voice messages, and social media Apps on smartphones act as the medium of such transmissions. When developing smartphone or mobile forensic investigation guidelines and recommendations, law enforcement agencies must take great care when it comes to steganography in mobile devices [5].

Today's most popular smartphones operate with Apple iOS or Android operating systems, all of which require customer interaction through Apps [6] [7]. Extracting evidence from an Android smartphone application requires extensive knowledge and experience with mobile operating systems (OS) and their file structures. "Android uses packages known as Android Package Kit (APKs) to arrange code in an application that also manages that application. An APK file is a package of all parts to a specific program. It contains the program code, resources, assets, certificate, and manifest file. In general, there is a pre-defined file structure for programming code and its resources organized into several folders, those being src/, gen/, libs/, res/, and assets/" [8].

Of recent, there are smartphone apps that are powered by artificial intelligence (AI) [9], [10]. AI does not just do one thing repetitively; it is intelligent and can do more by adapting to situations. Although AI is run by algorithms, it is more "human-like" than traditional machine learning (ML). Google's Android OS uses AI to better Apps such as Google Assistant, which allows for two-way communication. AI-powered Apps on smartphones often study user's device usage like news reading choices, behavior, and interests in order to learn and fine-tune their algorithms. However, there are privacy issues and concerns [11], [12] on AI being the technological rave these days. For example, apps such as the FaceApp which uses AI to process a picture of your face (into your older-self or change your face to the opposite sex) comes with privacy and security concerns over the Apps permission settings, data storage location (on servers based in a foreign country or cloud) and the company's often irrevocable rights to user data that is sandwiched somewhere in their user agreements. AI-powered Apps, while useful, maybe too smart for our own good.

Popular Apps like FaceApp [13] and TikTok [14] are a craze across the world. TikTok and FaceApp have been downloaded by millions of users [15], [16] across Android and Apple Play stores. FaceApp offers image transformations that have grown in popularity and TikTok offers music-infused virility on the social network platform by helping share short lipsync, comedy, and talent videos. Most of TikTok's Western user-base originally belonged to Musical.ly while its Eastern audience still uses a separate version of the App in China called Douyin. TikTok employs artificial intelligence (AI) to display personalized content to the user by analyzing user interests and preferences through their interactions with the content on the App [17]. While their popularity has grown tremendously in the last three years, concerns about their impact on users and society have been raised [18], [19], [20], [21]. TikTok addiction is also becoming a concern [22].

TikTok is owned by the Chinese Internet technology company "ByteDance" and was recently under investigation by UK authorities for potential mishandling of children's data [23]. While, ByteDance has made some changes since being dragged to various courts across geos, privacy concerns continue to be raised since its business relies on user-generated content. FaceApp is owned by a Russian company "Wireless Lab" and has attracted criticism of possible data misuse and privacy violations [24], [25].

To investigate the above concerns and take a deep dive into these Apps, the authors analyze the code behind these Apps and evaluate their robustness against steganographic attacks, privacy, and security. Both these Apps are subject to a series of tests that evaluate their security and forensic stance. Lastly, their end-user license agreement is discussed around privacy, copyright, ownership of user data. Since, these Apps have not yet been thoroughly investigated from various viewpoints, the authors propose a methodology through which the analysis of Android App artifacts can be assessed for their security, privacy, data management, and forensic postures.

## 2. RELATED WORK

Reverse engineering of application code has long been performed to try and study the code logic, architecture, and design. From reverse engineering an application, the code design can be reconstructed to a certain degree. Subsequent App cloning by code-reuse and its repackaging have been of interest to malware authors. The openness of the Android App market and the lack of adequate security testing by App developers have led to malware writers to target these apps to create and spread smartphone malware. Gongalez et al. [26] have proposed AndroidSOO that targets an extractable attribute called "String Offset Order" to detect such repackaging symptoms on Android Apps. While such detection is also possible by comparing App signatures with known signature databases, zero-day exploits can still go undetected when using the signature-based methods. Rastogi et al. [27] study in detail a few offline and online repackaging detection techniques that use different features and metrics to detect similarity of Apps. Lim et al. [28] propose a logic to check Apps for signs of application debugging while executing on a jailbroken or rooted device. Smartphone forensics tools check for malware as the investigator can take necessary precautions when encountering suspicious Apps.

However, traditional smartphone forensic tools struggle to detect anti-forensic approaches like steganography which can be a challenge for the forensic investigator when uncovering crime facts from smartphone evidence. Sporea et al. [29] study and test free and easy-to-use anti-forensic Apps available for smartphones and conclude them to be effective in their functionality. The availability of such Apps to trivialize the application of anti-forensics can be viewed as a dual-edge sword since novice malware authors or scamsters can take advantage of these Apps to hide sensitive information and share on social media. Android Play Store and Apple Store have increasingly enforced security checks on their portfolio of available Apps for security backdoors. Zhou et al. [30] proposed an app similarity measurement system called DroidMOSS and their experiments concluded that 5% to 13% of Apps hosted on various marketplaces were repackaged. Balebako et al. [31] conduct a series of interviews with 13 App developers on privacy and security decision-making and conduct an online survey of 228 App developers to

verify if privacy and security behaviors implemented during App development are related to characteristics of the App development companies. Their findings suggest that smaller development companies are less likely to demonstrate positive privacy and security behaviors and App developers were not aware of data collected by third-party tools for ads and analytics running on their Apps. While security-driven App development may still be loosely implemented, Apps often store user data in the cloud causing additional forensic challenges. Krishnan et al. [32] highlight the privacy and legal challenges when working with the cloud leading to forensic investigation limitations. To make things complicated, the mix of Artificial Intelligence (AI) and low-cost storage on the cloud results in huge volumes of data that is difficult to access for the forensic investigator. Little known to the lay smartphone user, AI-driven Apps and cloud may not seem an important issue, but their privacy is at risk. Solanas et al. [33] explain how AI can also assist in enhancing security and protecting privacy. There is however a literature gap in understanding the privacy demerits of AI coupled with cloud and the craze of video-sharing platforms on smartphones. To address these gaps, the authors examine two popular smartphone Apps (FaceApp and TikTok) on their coding practices, security posture, anti-forensics, and privacy.

### 3. TEST ENVIRONMENT

The research experiment was conducted to address a few areas of interest like reverse engineering of the App files, security, and steganography checks. A separate parallel study of the end-user agreement was also undertaken. Table I lists the model of smartphone and App versions, while, Table II lists the accompanying software used for the experiment. An online product research, cost and prior tool experience helped decide the software applications that were used in the experiment. The Apps were downloaded from the Android App store and were not activated with any subscription payments in order to mimic their general user base.

Smartphone	Motorola Nexus 6
Role in Experiment	Primary device for FaceApp and TikTok Apps
Sim Card	none
Android OS version	7.1.1
Phone Storage	32 GB
Access Level	Root
Build Number	N6F27M
Smartphone	Samsung Galaxy Note 3
Role in Experiment	Secondary device with TikTok App. Used to receive shared TikTok videos from Motorola Nexus 6
Sim Card	none
Android OS version	5.0
Phone Storage	32 GB
Access Level	Root
Build Number	LRX21V.N900KKKU0GP11
Forensic Computer	64-bit Windows 7 SP1, Intel Core i3, 8 GB RAM

**TABLE I:** Smartphone and Computer Hardware Used.

To perform steganography tests, image and video files were edited with secrets and uploaded into the Apps (FaceApp and TikTok) utilizing an Internet connection. The images and video files were then downloaded from the Apps for analysis. The End User License Agreement (EULA) was analyzed for privacy and copyright details from the EULA found on the Google Play store and from the company websites.

Software	Version
Nexus Root Toolkit	2.1.1
Root Checker Basic	6.4.6
Wireshark	3.0.5
Network Miner free version	2.4
StegoMagic	1.0
Dex2jar	2.0
JD-GUI	1.6.3
Paraben E3: DS Mobile Forensics	2.1.11597.15639
DB Browser (SQLITE)	3.11.2
Steghide	3.0
Java	12.0
Android Debug Bridge (ADB)	1.0.40
FaceApp	3.4.10
TikTok	12.3.5
IrfanView 64-bit	4.52
Androlyze	1.0

**TABLE II:** Software Tools used for In The Experiment.

#### 4. METHODOLOGY

In this section, the authors describe the environment used to examine the two Apps (FaceApp and TikTok). The goal of this experiment was multi-fold and the Internet was accessible for the smartphones via a router configured as a Wi-Fi Access Point. To verify the research questions of this experiment, the following hypotheses were formulated.

**H1:** Low-level Digital Forensic data of interest can be extracted by reverse engineering the App APK code files.

**H2:** Network Forensic data of interest can be determined from sniffing network traffic of both Apps.

**H3:** Data hidden (obfuscated) via steganography techniques on video and image files and then uploaded to the Apps will be altered during their processing.

**H4:** Random malware related text/strings when embedded in the metadata of image and video files will be removed when processed by these Apps.

**H5:** The Privacy Agreements of the App companies do not fully guarantee user privacy when using the Apps leaving room for concern.

At the very onset of the experiment, the smartphones were factory-reset to remove any previous data and the default settings were selected for the two devices. The primary smartphone earmarked for the Apps was rooted for this experiment using the Nexus Root Toolkit 2.1.9. The secondary smartphone was a Samsung Galaxy Note 3 that was used to receive media files when shared via TikTok from the primary Nexus phone. The experiment was divided into four stages;

one for reverse engineering the two Apps' code files, the second for steganography checks, the third for security posture, and the last for analysis of the EULA and Privacy Agreements.

The following methodology was followed in sequence for each stage. A Gmail account was created for the purpose of this experiment. The smartphone was set up using the newly created Gmail account and was used to access the Google Play store in order to download and install the two Apps, FaceApp and TikTok respectively. Default settings were chosen for the App installations. After the successful setup of the smartphone, a set of images and videos were identified for the experiment. The metadata of images and videos were edited to add malware names and related data that should raise some sort of security flags when these images are processed. Similarly, on the images and videos, hidden strings values were incorporated using steganography tools.

**Preparing the files for steganographic process:** For FaceApp, 12 .jpg images were identified along with 2 videos 15-second of .mp4 format for TikTok. Few unique strings/keywords were pre-identified with Malware and Ransomware names, hashes, etc. to see if they trigger a security flag on the back-end cloud servers that process the images and videos of FaceApp and TikTok. Using the Steghide command line tool and StegoMagic, a secret message was embedded in the image and video files respectively using steganographic techniques.

**For embedding steganographic information behind the .jpg files,** the following command was used in steghide command-line tool:

```
steghide embed -cf image1.jpg -ef secret.txt
```

**For embedding steganographic information behind the .mp4 files,** the following steps were performed in the StegoMagic tool as shown in Figure 1.

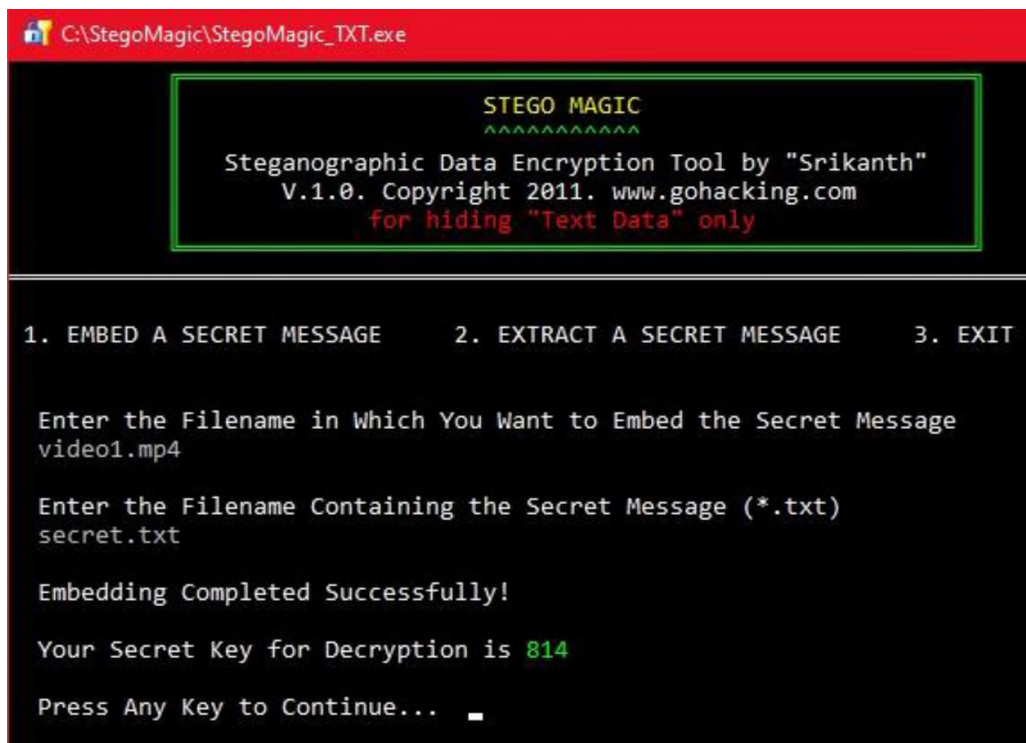


FIGURE 1: The process of embedding secret message behind the .mp4 file.

Furthermore, from the forensic workstation, the malicious malware and ransomware names were then added to these 12 images and 2 video files using editing tools like IrfanView and Windows File Explorer. Lastly, the files were copied into the smartphone using the forensic workstation in order for them to be processed and used by the Apps.

The Nexus and Samsung smartphones were configured to a wireless access point (WAP). A wireless router working as a WAP was used for the two smartphones. A D-Link switch supporting port mirroring was used to connect the WAP, monitoring PC, and the Internet. Port mirroring was enabled on the WAP port of the switch such that all traffic passing through the WAP was mirrored to the monitoring PC. Once this design was confirmed as working, Wireshark capture was started on the monitoring PC. FaceApp and TikTok Apps were downloaded and installed on to the Nexus smartphone. They were opened for use and closed. Wireshark capture was stopped, and traffic captured was saved for later analysis. Only TikTok was downloaded on the Samsung smartphone as it would be used only to receive any videos posted via TikTok on the Nexus smartphone.

**Packet capture for FaceApp:** Wireshark captured was started and the identified images with edited metadata were uploaded to the Nexus smartphone. FaceApp was opened on Nexus and the images were submitted without applying any filters. The choice of not using any predefined filters of the App was to avoid filters working on the previously hidden messages behind the images. Wireshark capture process was stopped, and the packet capture was saved as a .pcap file for later analysis. Images post submission to FaceApp then were downloaded to a forensic PC for steganographic analysis.

**Packet capture for TikTok:** Wireshark captured was started and the videos (previously edited with StegoMagic), were shared via TikTok to the public using unique hashtags. The Samsung smartphone was used to check for posted these videos via TikTok on the Nexus smartphone. No predefined editors of TikTok were used to try and mimic a realistic scenario of a malicious actor trying to share videos with hidden messages to a recipient (another TikTok user). Wireshark capture was stopped, and the packet capture was saved as a .pcap file for later analysis.

**For reverse engineering the two Apps,** Androguard was used. Using Androguard, the authors were looking for any security flags, privacy, ownership, and general programming discipline like class/method/variable naming conventions, meaningful comments, exception/garbage handling etc. Androguard modules such as Androlyze, Androdd, Androapkinfo, Androrisk, Androsign, and Androxxmml used for analysis. To convert classes.dex files to classes.jar, dex2jar application was used. JD-GUI App was then used to view the Java classes of the Apps.

The smartphone was also left powered-on for a day without use. After a day, the smartphone and registered Gmail account was checked for any warnings or device alerts from the two Apps.

Paraben was used on the forensic workstation to acquire the Android smartphone using physical extraction methods. Both the Apps file locations were identified on the Paraben case and the 12 images and 2 video files were identified. These files were reprocessed by steghide and StegoMagic to check for the secret string. Analysis of the App files was carried out using dex2jar and JD-GUI. The Apps database files were analyzed using the DB Browser (SLLITE) tool. Metadata of the images and 2 video files were also rechecked for the previously embedded strings.

#### **A. EULA and Privacy Agreements**

The authors performed a deep dive into published policies on the company websites of FaceApp and TikTok. The authors also reviewed the policies that a user accepts when installing the Apps on his smartphone and also the policies available on the Apps' menus.

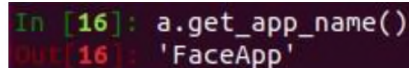
## 5. ANALYSIS and RESULTS

In this section the authors analyze results from the above experiments and explain their findings. Both the Apps (FaceApp and TikTok) were installed on the Nexus smartphone starting with FaceApp.

### 5.1 Reverse Engineering

Androguard [34] is an open source Python-based tool used for reverse engineering the Android-based applications. Androguard takes the application's Android package (.apk) files as an input and breaks it down for analysis. Androguard can be installed on Windows, Linux, and OSX with Python version 3.4 or above as a pre-requisite. In this research Ubuntu Linux distribution [35] was used as a platform for Androguard to perform the analysis of FaceApp and TikTok. The following command can be used to install the Androguard on Ubuntu.

**sudo apt install androguard**



```
In [16]: a.get_app_name()
Out[16]: 'FaceApp'
```

FIGURE 2: Method to get the App name.

To start the analysis, **androlyze -s** command was used in the terminal of Ubuntu. This command opens an iPython shell that includes all the modules required to perform an analysis.

To start the analysis, AnalyzeAPK(filename) function was loaded with the .apk file as shown below. In this case, the FaceApp apk [36] file was used for the analysis. This .apk file was downloaded on 09/13/2019 at 10:27 pm US Central Time.

**a, d, dx = AnalyzeAPK("Downloads/FaceApp v3.4.14 apkpure.com.apk")**

The three objects are, **a an APK object**, **d an array of DalvikVMFormat**, and **dx an Analysis object** respectively. APK object contains get\_app\_name() method to get the App name as shown in Figure 2.

All the permissions in the FaceApp application was obtained by using the get\_permissions() method of an APK object. FaceApp has the permission to access the internet, access the network state, access camera, reading, and writing to the external storage. Figure 3 shows the list of all the permissions on the FaceApp App code.



```
In [3]: a.get_permissions()
Out[3]:
['android.permission.INTERNET',
'android.permission.ACCESS_NETWORK_STATE',
'android.permission.CAMERA',
'android.permission.WRITE_EXTERNAL_STORAGE',
'android.permission.READ_EXTERNAL_STORAGE',
'com.android.vending.BILLING',
'com.google.android.providers.gsf.permission.READ_GSERVICES',
'android.permission.WAKE_LOCK',
'com.google.android.c2dm.permission.RECEIVE',
'com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE']
```

FIGURE 3: All the Permissions in FaceApp.

The APK object also includes get\_activities method to list all the activities in the application. A few of the listed activities are Facebook Activity, Custom Tab Activity, Main Activity, and Matisse Activity. Figure 4 shows all the activities associated with the FaceApp application.



```
In [8]: a.get_activities()
Out[8]:
['com.facebook.FacebookActivity',
'com.facebook.accountkit.ui.AccountKitActivity',
'com.facebook.CustomTabActivity',
'io.faceapp.MainActivity',
'com.zhihu.matisse.ui.MatisseActivity',
'com.facebook.accountkit.ui.AccountKitEmailRedirectActivity',
'com.facebook.CustomTabMainActivity',
'com.google.android.gms.auth.api.signin.internal.SignInHubActivity',
'com.google.android.gms.common.api.GoogleApiActivity',
'com.google.android.gms.ads.AdActivity',
'com.android.billingclient.api.ProxyBillingActivity',
'com.jakewharton.processphoenix.ProcessPhoenix']
```

FIGURE 4: All the Activities in FaceApp.

The APK object has `get_signature()` method that returns the data of the first signature file found which can be either v1 signature or JAR signature. Figure 5 shows the signature returned from the apk.

```
In [7]: a.get_signature()
Out[7]: b'\x02\x04;\x06\t*\x86H\x86\xf7r\x01\x07\x02\xa0\x82\x04,0\x82\x04(\x02\x01\x01\x06\t*\x86H\x86\xf7r\x01\x07\x01\xa0\x82\x02\xbf0\x82\x02\xbb0\x82\x01\xa3\xa0\x03\x02\x01\x01\x0b\x05\x000r1\x0b0\t\x06\x03U\x04\x06\x13\x02RU0 \x17r170201154402Z\x18\x0f2062\x01"0r\x06\t*\x86H\x86\xf7r\x01\x01\x05\x00\x03\x82\x01\x0f\x000\x82\x01\n\x02\x8\x92\xe5\x92\x8f\xdb0\xb3\xfc\x0b"\xb0\x01\xfa\x11\xca\x04\xe20\x0f\x05\x88\xf4w5\xf3\x1'\x07\xe2\x06\xa0\x8e_\x17\xfc\x194\x07\x03r\x0d9\x8f\x128`rK\xb5\x06\x0f34\xb6\x8eH0\xe\x956\x86\xad\x0e\xb8\xe8\x97\x90\xfe\xafI\x04\x07=\x07\x04\x0e\x1b\x18\xa8\xba\x0f3\t\x068c\xfe\x066\x05\x0b\x8e\x17Mr\x0e\x80\xbb<\x140#%\x07\x89~"\x03\x05\x0fM\x0d\x09\xbb\x0\xad5~\x0a\x09d\x0d\x00.\xb8\x9b\x05\x06r\x0a1<"qg\x03R,\x03\x06|tMk\x07\x0c\x01\x0d\x06\x03U\x0d\x0e\x04\x16\x04\x14\x01\x06\x8e\xfb\x04\x1c\x88\x08\x0d|\x09\x06\x0b\x01\x0b\x05\x00\x03\x82\x01\x01\x00t\x0f*\x03\x01\x0e5\x18"\xe4\x03\x0f2\x08\x11\x03\x05\x00\xe3\x0f7/\x01a\x88t\x0b\x07\x04\x18\x0fQ\x0e1\x8a\x0b7\x0ach\x09\x0b4\x07L\x0d)\x0b\x0a\x0d\x0e\x01\x07f|+\x0d7\x02r\x0b\x099l\x02}0'9'\x88\x84y\x0d\x0a5\x0e\x0f\x08\x0d\x0a\x0aX0N\x14\x13877\x87\xa9\x0f\x0f0\x06\xa05\x0e|\x0f2lb\x0e1\x8e\x0d\x0b4-0\x0f\x03~\x0t\x0d8\x0d\x15r\x0e6h\x0q\x0c8\x09\x0b\x0b\x0e7#\x03\x05\x0e1r\x0f\x01eb\x04\x13Zn` \x0b\x0b\x0e5\x0e\x0c9\x093B{*. "\x03rb6\x1a\x0d9\x0c\x0a\xa0\x13\x0f1\x0b\x0f\x0cb^\x0d,\x0a\x02\x0een\x09\x10\x0c6\x0e7\x0d\x0b\x09a\x0f\x0d6\x03U\x04\x06\x13\x02RU\x02\x04I\x0db\x0d\x0e10r\x06\t*\x86H\x01e\x03\x04\x02\x01\x05\x00\x01\x00\xa4\x06nX\x13\x098\x0f4\x0fa\x06u\x0a9E\x08m\x0c2\x138\x02\x0b\x0f\x09dU\x0c\x0d2\x0e\x0f\x0b3o\c4\x02\x0c7\x083\x09a?\x0b7\x13ps0m?\xa6\xa5\x0f2\x0c\x82\x094U\x0acv\x0e5\x087\x0e8%\x01\x0da>\x0a5KR#\x0e1\x0c5\x0e94\x0be+^\x09ae\x0d\x0a3\x0f2\x0c1\x0b\x0a\x0e3\x0f1\x0d5\x09e\|0.\x0b\x0818v\x0d\x0a12\x09\x0d6\x01\x0c7\x0e1>\x0b8p\x11MK\x0b3k\xa9\x08\x0d9GW\x0f9\x0b9\x0b0\x02\xa3\x0ca\x08d>Bf\x0b\x0d\x09f7x01q\x0fbo\x083\t|\x089T\xa6(1D\xa1B\x096\x0d9R\x0acm$\x05\x09c\x07f\x0d4\x09f\x0d6\x0c1M\x09c\x06\x11\x082\x0e\x0c7\x0a5\x02'
```

FIGURE 5: Signature in FaceApp.

Figure 6 shows the name of the signature file of the FaceApp application. Figure 7 shows the details of each permission found in the FaceApp. The `get_details_permissions()` method return the details of all permissions. These details include the protection level, label, and description of each permission. APK object contains `get_signature_name()` that returns the name of the first signature file found.

```
In [14]: a.get_signature_name()
Out[14]: 'META-INF/CERT.RSA'
```

FIGURE 6: Signature name of FaceApp.

To start the analysis of the TikTok [37] application, `AnalyzeAPK(filename)` function was loaded with the .apk file as shown below. This .apk file was downloaded on 09/13/2019 at 10:29 pm US Central Time.

```
a,d,dx=AnalyzeAPK("Downloads/TikTok Make Your Day v13.0.3 apkpure.com.apk")
```

```
In [10]: a.get_details_permissions()
Out[10]:
{'android.permission.ACCESS_NETWORK_STATE': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.CAMERA': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.INTERNET': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.READ_EXTERNAL_STORAGE': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.WAKE_LOCK': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.WRITE_EXTERNAL_STORAGE': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'com.android.vending.BILLING': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'com.google.android.c2dm.permission.RECEIVE': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'com.google.android.providers.gsf.permission.READ_GSERVICES': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference']}
```

FIGURE 7: Details of Permissions in FaceApp.

The three objects are **a** an **APK object**, **d** an **array of DalvikVMFormat**, and **dx** an **Analysis object** respectively. APK object contains get\_app\_name() method to get the App name as shown in Figure 8.

```
In [28]: a.get_app_name()
Out[28]: 'TikTok'
```

FIGURE 8: Method to get the App name.

All the permissions in the TikTok application were obtained by using the get\_permissions() method of an APK object. TikTok has the permission to access the Internet, access the network state, access camera, reading, and writing to the external storage. Figure 9 shows the list of few permissions on the TikTok App code.

```
In [3]: a.get_permissions()
Out[3]:
['android.permission.INTERNET',
'android.permission.ACCESS_NETWORK_STATE',
'android.permission.READ_EXTERNAL_STORAGE',
'android.permission.WRITE_EXTERNAL_STORAGE',
'android.permission.ACCESS_WIFI_STATE',
'android.permission.CAMERA',
'android.permission.RECORD_AUDIO',
'android.permission.FLASHLIGHT',
'android.permission.WAKE_LOCK',
'android.permission.GET_TASKS',
'android.permission.READ_CONTACTS',
'android.permission.RECEIVE_BOOT_COMPLETED',
'none.used.ACCESS_FINE_LOCATION',
'none.used.ACCESS_COARSE_LOCATION',
'android.permission.VIBRATE',
'com.meizu.c2dm.permission.RECEIVE',
'com.zhiliapp.musically.permission.READ_ACCOUNT',
'com.zhiliapp.musically.permission.WRITE_ACCOUNT',
'android.permission.REQUEST_INSTALL_PACKAGES',
'com.android.launcher.permission.INSTALL_SHORTCUT',
'com.android.launcher.permission.UNINSTALL_SHORTCUT',
'com.android.launcher.permission.READ_SETTINGS',
'com.android.launcher.permission.WRITE_SETTINGS',
'com.android.launcher2.permission.READ_SETTINGS',
'com.android.launcher2.permission.WRITE_SETTINGS',
'com.android.launcher3.permission.READ_SETTINGS',
'com.android.launcher3.permission.WRITE_SETTINGS',
'android.permission.AUTHENTICATE_ACCOUNTS',
'com.htc.launcher.permission.READ_SETTINGS',
```

FIGURE 9: List of few permissions on TikTok.



The APK object also includes get\_activities method to list all the activities in the application. A few of the listed activities are Follow Follower Activity, Avatar Choose Activity, Avatar Cut Activity, and Welcome Screen Activity. Figure 10 shows all the few of the activities associated with the TikTok application.

```
In [30]: a.get_activities()
Out[30]:
['com.ss.android.ugc.trill.openauthorize.AwemeAuthorizedActivity',
'com.ss.android.ugc.aweme.i18n.musically.follows.FollowFollowerActivity',
'com.ss.android.ugc.aweme.i18n.musically.profile.ui.MusHeaderDetailActivity',
'com.ss.android.ugc.aweme.i18n.settings.privacy.MusPrivacyActivity',
'com.ss.android.ugc.aweme.i18n.settings.agreements.AgreementActivity',
'com.ss.android.ugc.aweme.i18n.checkprofile.CheckProfileActivity',
'com.ss.android.ugc.aweme.i18n.settings.language.MusChooseLanguageActivity',
'com.ss.android.ugc.aweme.i18n.settings.blacklist.MusBlackListActivity',
'com.ss.android.ugc.aweme.i18n.musically.cut.AvatarChooseActivity',
'com.ss.android.ugc.aweme.i18n.musically.cut.AvatarCutActivity',
'com.ss.android.ugc.aweme.setting.ui.MusSettingNewVersionActivity',
'com.ss.android.ugc.aweme.setting.ui.MusSettingManageMyAccountActivity',
'com.ss.android.ugc.aweme.sticker.prop.activity.StickerPropDetailActivity',
'com.ss.android.ugc.aweme.notification.newstyle.MusNotificationDetailActivity',
'com.ss.android.ugc.aweme.notification.newstyle.MusFollowRequestDetailActivity',
'com.zhiliaoapp.musically.openauthorize.AwemeAuthorizedActivity',
'com.ss.android.ugc.aweme.welcome.WelcomeScreenActivity',
'com.ss.android.ugc.aweme.interest.InterestSelectActivity',
'net.openid.appauth.RedirectUriReceiverActivity',
'com.ss.android.ugc.aweme.app.AppLinkHandler',
'com.facebook.accountkit.ui.AccountKitActivity',
'com.ss.android.ugc.aweme.share.SystemShareActivity',
'com.ss.android.ugc.aweme.splash.SplashAdActivity',
'com.ss.android.ugc.aweme.feedback.FeedbackActivity',
'com.ss.android.ugc.aweme.feedback.SubmitFeedbackActivity',
'com.ss.android.ugc.aweme.setting.ui.MusSubmitFeedbackActivity',
'com.ss.android.ugc.aweme.discover.activity.DiscoverDetailActivity',
'com.tencent.tauth.AuthActivity',
```

FIGURE 10: Few Activities of TikTok.

Figure 11 shows the details of each permission found in the TikTok. The get\_details\_permissions() method return details of each permission. The details include the protection level, label, and description of each permission. APK object contains get\_signature\_name() that returns the name of the first signature file found.

```
In [32]: a.get_details_permissions()
Out[32]:
{'android.permission.ACCESS_NETWORK_STATE': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.ACCESS_WIFI_STATE': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.AUTHENTICATE_ACCOUNTS': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.BROADCAST_STICKY': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.CAMERA': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.FLASHLIGHT': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.FOREGROUND_SERVICE': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
```

FIGURE 11: Permission detail in TikTok.

[illegible]

**FIGURE 12:** Signature in TikTok.

Figure 12 shows the name of the signature file of the TikTok application. APK object has `get_signature_name()` method that returns the data of the first signature file found which can be either v1 signature or JAR signature. Figure 13 shows the signature returned from the apk.

```
In [34]: a.get_signature_name()
Out[34]: 'META-INF/CERT.RSA'
```

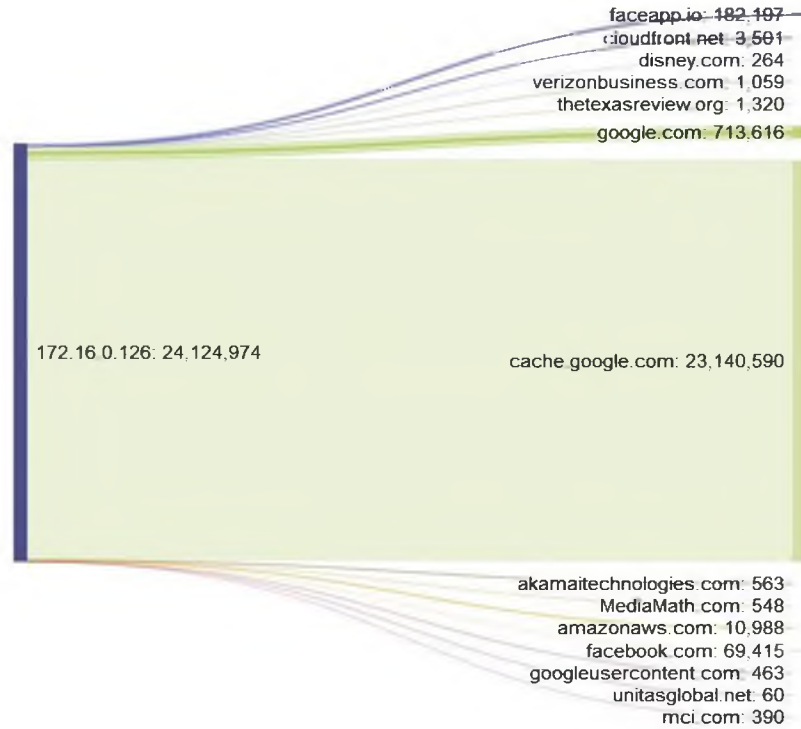
**FIGURE 13:** Signature name of TikTok.

Users' personal and financial information is tied to the rise in the use of mobile applications. The security of users' data should be of utmost importance for an application developer. This is the reason why the area of reverse engineering is gaining momentum and tools such as Androguard help in forensic investigation. The contribution of Androguard towards mobile forensic research is twofold. It can be used to reverse engineer mobile applications' code to check for any security vulnerability. Lastly, it can detect changes in applications' code packaging in order to check the integrity of the application.

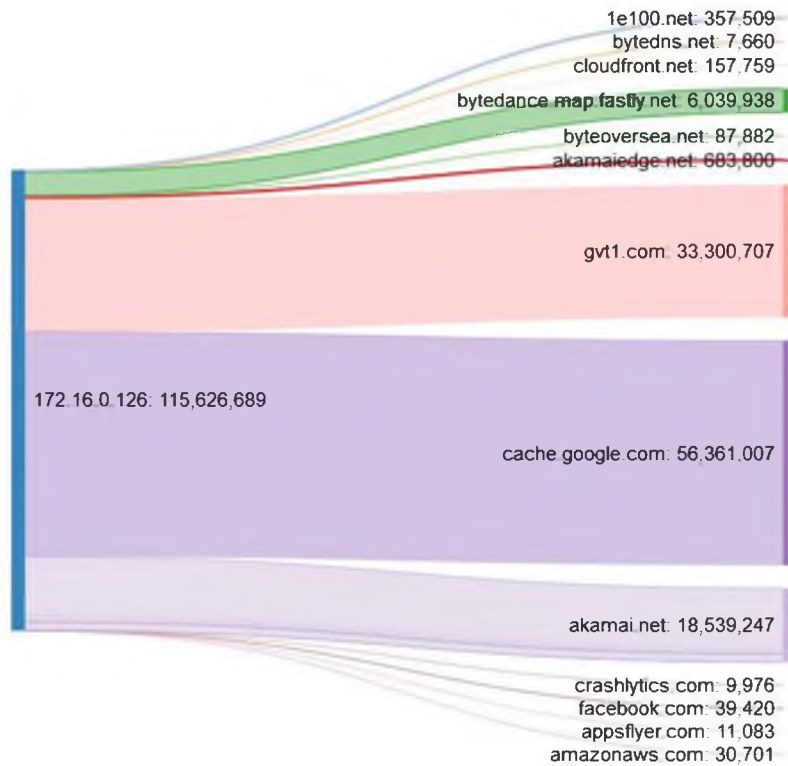
## 5.2 Network Traffic

On analysis of all .pcap files from Wireshark captures using port-mirroring, the authors conclude that FaceApp uses encryption for all traffic. Thus, it would be difficult for a network sniffer to deduce the traffic from FaceApp.

On the other hand, the authors discovered that the videos and thumbnail images received from TikTok on the Samsung smartphone were left unencrypted and can be easily sniffed on the network. This can potentially compromise the privacy of the user as a malicious sniffer can easily deduce the videos surfed by the TikTok user. However, when a video was uploaded from the Nexus smartphone, network traffic to TikTok servers was encrypted. Figure 14 and Figure 15 show the network traffic patterns of these Apps during their download and install on the Nexus smartphone. Figure 16, Figure 17 and Figure 18 shows the videos and images that were received on the Samsung smartphone as unencrypted.



**FIGURE 14:** Sankey Diagram depicting network traffic of FaceApp during download from the Android play-store and install process.



**FIGURE 15:** Sankey Diagram depicting network traffic of TikTok during download from Android play-store and install process.



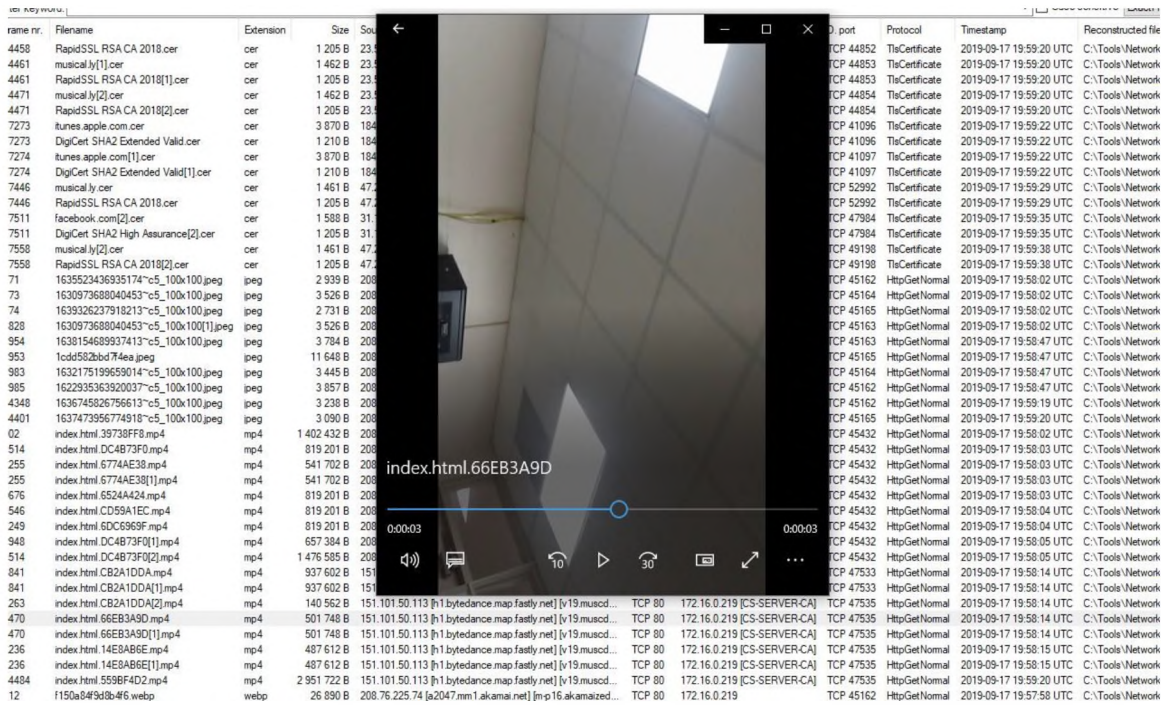


FIGURE 16: Videos reassembled from TikTok network traffic using Network Miner.

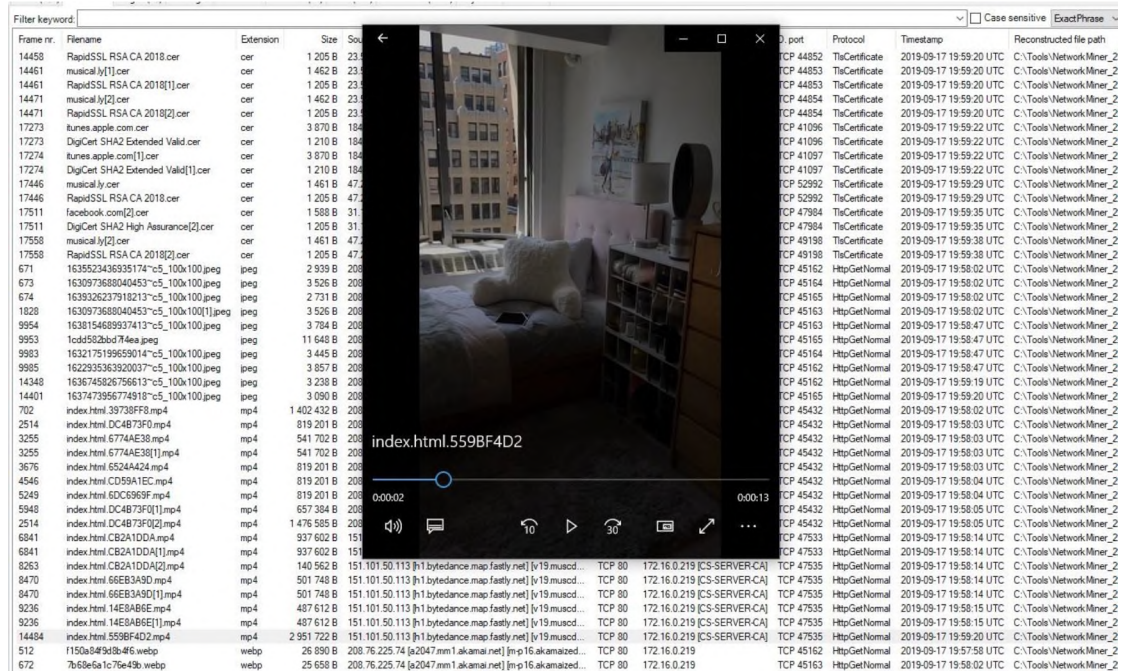


FIGURE 17: Videos reassembled from TikTok network traffic using Network Miner.

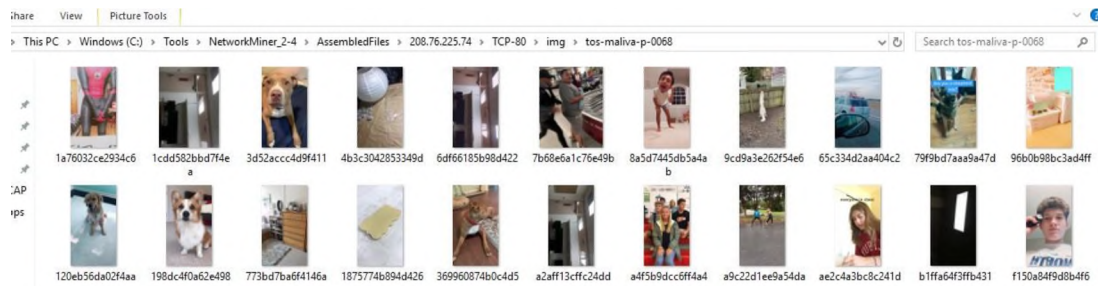


FIGURE 18: Thumbnail images reassembled from TikTok network traffic using Network Miner.

### 5.3 Steganography

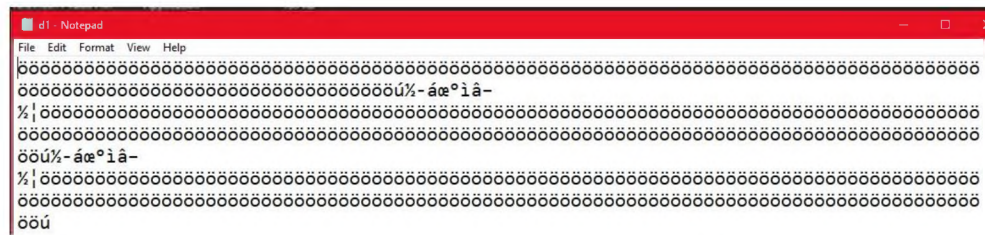
StegoMagic and Steghide were used for steganographic analysis. Stego images and .mp4 videos were used in FaceApp and TikTok. With FaceApp, images with hidden text message information were uploaded for processing by FaceApp. No filters from the FaceApp options were applied as this would probably be the best way for an image with hidden-text to be processed without FaceApp AI algorithms interference. Alternatively, if a predefined filter was applied, it would most certainly allow for AI to be applied by the FaceApp algorithms. The result after FaceApp upload and processing was a resulting image without any hidden-text. The authors could conclude that FaceApp did indeed strip out the hidden-texts by applying anti-stego checks.

For TikTok, a hidden message text file was embedded behind the two .mp4 files and was shared via hashtags with the public from the Nexus smartphone. The TikTok app on the Samsung smartphone received the publicly shared .mp4 files. They were analyzed for the previously stored hidden message text. No such hidden texts were found on these .mp4 videos. On the source phone, i.e. the Nexus phone, the hidden text files were recovered via StegoMagic after saving it back to the forensic workstation. But the actual text in the hidden message file was garbled. The authors conclude that the .mp4 files were stripped off their hidden texts by the TikTok algorithms on the receiving phone. This would greatly hamper malicious actors from sharing the steganographic information on this platform. Figures 19 and 20 show the hidden message extraction process and garbled file obtained from the .mp4 files on source Nexus phone respectively.



FIGURE 19: The Extraction Process in StegoMagic.





**FIGURE 20:** The garbled text file extracted from the Nexus phone.

## 5.4 Security Posture Evaluation

To ascertain security posture FaceApp and TikTok, metadata of selected images, and .mp4 files were edited with texts that would raise low-level security flags. Texts pertaining to malware names, popular threat actors etc. were inserted within the metadata of these images and .mp4 files. Both FaceApp and TikTok stripped away such metadata from the images and .mp4 files. The authors conclude that these Apps do possess low-level checks against metadata strings to weed out irreverent details. This could hamper malicious actors from utilizing metadata fields of images and .mp4 to embed malicious programs or text.

## 5.5 Analysis of EULA and Privacy Agreements

FaceApp's privacy policy [38] is available to the public on the Internet. The below can be inferred from this policy.

## Information Collection

1. The FaceApp company collects user content (e.g., photos and other materials) that users post through the App service (communications between user and FaceApp).
2. Third-party analytics tools are used by the FaceApp company to measure traffic and usage trends for the Service.
3. The smartphone's browser log file information is automatically reported to the FaceApp company each time the user makes visits to a web page or App.
4. FaceApp "might" store the photos/image users have chosen to upload in the cloud for a short period of time to enhance "performance and traffic" and to avoid repeat bursts of uploads of the same image file.
5. FaceApp's privacy policy clarifies that the App takes additional data like user location, IP address, and log file information for the purpose of offering targeted ads at the user.
6. On children's privacy, FaceApp policy states that it does not knowingly collect or solicit any information from anyone under the age of 13.

## Information Usage

1. Online ads or other forms of marketing can be personalized and delivered to the user and others.
2. Provide, improve, test, and monitor the effectiveness of our Service.
3. Develop and test new products and features.
4. Monitor metrics such as total number of visitors, traffic, and demographic patterns.
5. Information Sharing.
6. FaceApp will not rent or sell user information to third parties outside FaceApp without user consent, except parties with whom they may share our information.
7. FaceApp may share user content and user's information (including but not limited to, information from cookies, log files, device identifiers, location data, and usage data) with businesses that are legally part of the FaceApp.
8. FaceApp may remove parts of data that can identify a user and share anonymized data with other parties.



9. Parties with whom the user may choose to share the user content that the user voluntarily processes with the service, becomes available to the FaceApp anonymously.
10. FaceApp may remove parts of data that can identify you and share anonymized data with other parties.

FaceApp boasts of implementing AI-powered selfie-editing and their privacy agreements are found in detail on their website on the Internet. However, the average user of FaceApp seldom visits and reads the privacy agreement contents detailed on the FaceApp website. Users often click through such privacy agreements and notifications. They are also seen to cover the company's potential liabilities. FaceApp processes all its images on its servers and not on the smartphone. A user cannot override the App settings to decline access to media storage. The App needs access to device storage and the Internet to function. Such a design and the company's decision to store uploaded images/photos on their servers for a short period of time raises few controversies over privacy and ownership. On the ownership and copyrights front, according to FaceApp, users can request their data to be deleted via the mobile App using "Settings → Support → Report a bug" with the word "privacy" in the subject line. All FaceApp features are available without the user logging in and thus FaceApp doesn't have access to any data that could identify a person against the uploaded images/photos. FaceApp has gone great lengths to communicate its privacy and ownership policies with concerned users and media. The company has clarified that even though the core R&D team is located in Russia, the user data is not transferred to Russia.

From TikTok's publicly available and published privacy policy [39], the below can be inferred from this policy.

#### **Information Collection**

1. TikTok collects information such as user's contact details, the content they create, location, and credit card details.
2. TikTok collects information that the user shares with TikTok from third party social network providers, and technical and behavioral information about the user's use of the application.
3. TikTok collects information contained in the messages that users send through the application.
4. TikTok collects the user's contacts if they allow access to the phone book on the mobile device.
5. TikTok collects user's comments on the Platform or any other user-generated content and video content that users generate through and broadcast from the Platform.
6. TikTok collects the device IP address, location-related data, unique device identifiers, browsing history (including content the user have viewed on the App's platform), cookies, mobile carrier, time zone setting, and mobile or device information including the model of the device, screen resolution, operating system, App, file names and types, and platform.

#### **Information Usage**

1. To administer the platform (i.e. to provide the services to the user) and for internal operations, including troubleshooting, data analysis, testing, research, statistical and survey purposes (i.e. to guarantee the platform's stability and security).
2. To personalize the content the users receive and provide tailored contents that will be of interest to the users.
3. To improve and develop the platform and conduct product development.

### Information Sharing

1. TikTok shares information with business partners so that users receive special offers via the platform.
2. TikTok shares information with advertisers and advertising networks that require the data to select and serve advertisements.
3. TikTok shares information with its cloud storage providers for disaster recovery services.
4. TikTok shares information with its data centers and IT service providers.

Across countries like the USA, UK, India etc., TikTok continues to be heavily scrutinized by the concerned media, government regulators, courts, privacy watch agencies, and activists. While privacy concerns with TikTok's possible storage of data on servers in China are around, much of the privacy concerns seem to emanate from its video sharing platform and viewership. A major concern comes from its growing younger user base that is a lucrative target for the advertisement industry. Concerns over the collection of personal information of children, cyberbullying, exposing children to sexual predators, pornographic contents, etc. have also surfaced. With TikTok, a private account does not guarantee a child's profile photo, username, and other personal information from being visible to all TikTok users. The TikTok App features many privacy and safety settings to restrict who can contact and comment on a child's posts and profile. TikTok also features a Digital Wellbeing feature to restrict inappropriate content and help parents manage how long children spend on the App. TikTok also has a dedicated privacy policy for young users [40].

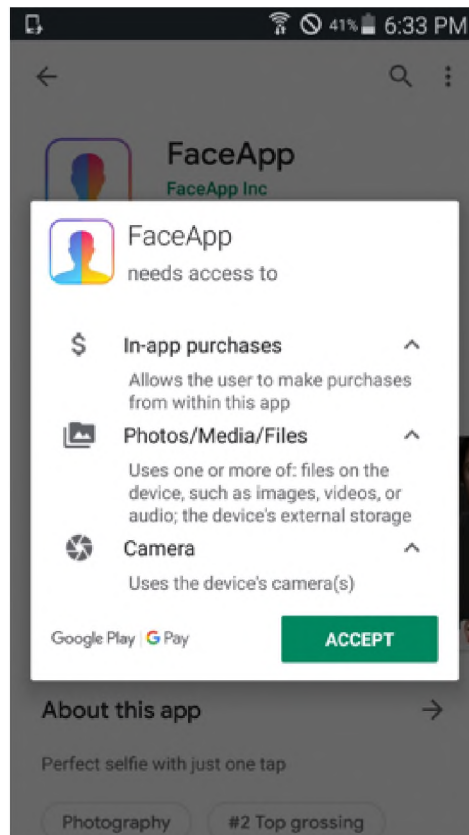


FIGURE 21: FaceApp Permission.

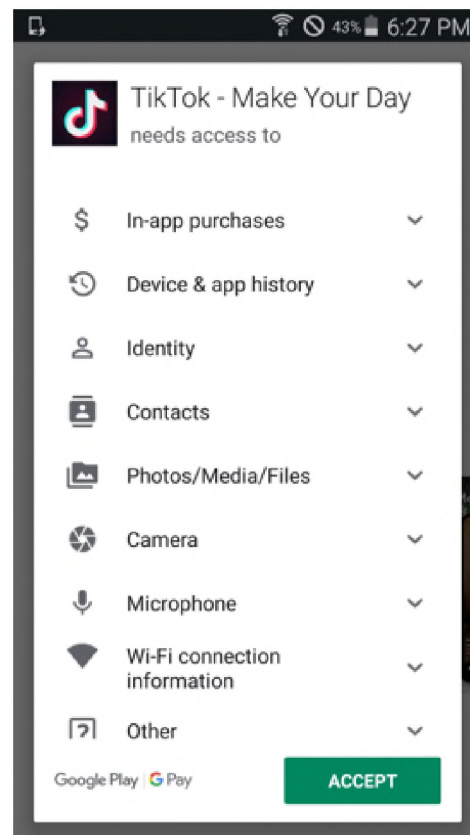


FIGURE 22: TikTok Permission.

## 6. CONCLUSION

Applications such as FaceApp and TikTok are currently enjoying huge success and are very popular across all ages with a huge spike in their downloads. Their strengths lie in their ease to use and the joy or fun that they bring along to the public user. In our evaluation of FaceApp for security hygiene and anti-stego checks, the authors found that the App did strip away implanted metadata and also removed hidden messages on the images when uploaded for processing by the App. These actions would impact malicious users and script-kiddies from using such Apps as a platform to share coded data. Steganography analysis has helped in determining that these apps strip out the hidden-texts from the uploaded the images that prevents malicious actors from sharing the steganographic information on these platforms. The application program code (.apk files) was inspected for both Apps to check for any malicious logic that can put a user's privacy at risk. By dissecting the .apk files, the authors found the hexadecimal signatures of the two apps, their RSA certificates, the user assigned permissions, and network activities to name a few. The analysis of these apps also involved the network traffic analysis that revealed that FaceApp uses encryption for all the traffic, however, TikTok doesn't encrypt the videos and thumbnails while transferring it from senders to receivers that makes it vulnerable for network sniffers. The authors recommend TikTok to encrypt all traffic irrespective of the type of user (paid or free). The reverse engineering of the APK files did show the various permissions that the App used and was found to be in-sync with the access permissions visible via the smartphone screen. The authors reviewed published privacy policies of both FaceApp and TikTok from the Internet and provided highlights while concurring that they seem to cover enough ground for a user. However, periodic reviews of their security posture, privacy, ownership, copyrights, and storage locations of these Apps are suggested as changes to them via newer versions or security patches can alter their architecture and design. As a best practice, the authors suggest the user community to ensure that no user sensitive or personal information is included within photos/images and videos when using such Apps. With facial recognition stirring up controversies coupled with deepfake technology, malicious actors can use publicly available photos to create stunningly realistic videos of a person using nothing more than a single image of their face, thereby increasing the probability of facial misuse.

In this paper, the authors evaluate FaceApp and TikTok Apps from different viewpoints to ascertain their security, privacy, data management, and forensic posture. The authors conclude that while it is important to have fun with technology, FaceApp and TikTok App users must think carefully about the safeguards they put in place to protect their identity, photos, videos, privacy, etc. while also considering the motives of these applications that promote such technology. As part of future work, the authors plan to investigate these Apps for their detailed forensic footprints and the extent to which they could support a forensic investigation.

## 7. ACKNOWLEDGMENT

The authors would like to thank the Cyber Forensics Intelligence Center at Sam Houston State University for providing forensic tools and software for undertaking this research.

## 8. DECLARATION

The author's analysis of copyrights, ownership, End User License Agreement (EULA), and privacy agreements of FaceApp and TikTok are for informational purposes only and not for the purpose of providing legal advice to readers. The opinions expressed on these subjects are that of the individual authors and may not reflect the opinions of the Department of Computer Science at Sam Houston State University or Cyber Forensics Intelligence Center at Sam Houston State University or that of Sam Houston State University.

## 9. REFERENCES

- [1] ANNUAL SURVEY HIGHLIGHTS – CTIA (2019, June 20). Retrieved from <https://api.ctia.org/wp-content/uploads/2019/06/2019-Annual-Survey-Highlights-FINAL.pdf>.

- [2] Demographics of Mobile Device Ownership and Adoption in the United States (2019, June 12). Retrieved from <https://www.pewresearch.org/internet/fact-sheet/mobile/>.
- [3] Goodison, S. E, Davis R. C. and Jackson B. A. (2015). Digital Evidence and the U.S. Criminal Justice System. Retrieved from <https://www.ncjrs.gov/pdffiles1/nij/grants/248770.pdf>.
- [4] Sun, H. M., Weng, C. Y., Lee, C. F., & Yang, C. H. (2011). Anti-forensics with steganographic data embedding in digital images. *IEEE Journal on selected areas in Communications*, 29(7), 1392-1403.
- [5] Burrows, C., & Zadeh, P. B. (2016, June). A mobile forensic investigation into steganography. In 2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security) (pp. 1-2). IEEE.
- [6] O'Dea S. (2020, February 27). Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 2nd quarter 2018. Retrieved from <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>.
- [7] Smartphone Market Share (2020). Retrieved from <https://www.idc.com/promo/smartphone-market-share/os>.
- [8] Allevato T. (2013). Structure of an Android Application. Retrieved from <http://sofia.cs.vt.edu/sofia-2114/book/chapter2.html>.
- [9] Easy Tech Trick (2018, July 21). Top15 Artificial Intelligence Apps For Android And iOS. Retrieved from <https://www.easytechtrick.org/artificial-intelligence-apps/>.
- [10] Arthur C. (2015, June 15). Artificial intelligence: don't fear AI. It's already on your phone – and useful. Retrieved from <https://www.theguardian.com/technology/2015/jun/15/artificial-intelligence-ai-smartphones-machine-learning>.
- [11] Cellan-Jones R. (2014, December 2). Stephen Hawking warns artificial intelligence could end mankind. Retrieved from <https://www.bbc.com/news/technology-30290540>.
- [12] FaceApp (n.d.) Retrieved from <https://faceapp.com/app>.
- [13] TikTok (n.d.) Retrieved from <https://www.tiktok.com/en/>.
- [14] Richter F. (2019, July 23). FaceApp Craze Reaches New Heights. Retrieved from <https://www.statista.com/chart/18769/estimated-worldwide-faceapp-downloads-by-platform/>.
- [15] Influencer Marketing (2019, April 14). 37 TikTok Statistics That Will Blow Your Mind. Retrieved from <https://influencermarketinghub.com/tiktok-statistics/>.
- [16] Xiao E. (2017, December 5). China's most addictive news app Toutiao eyes world domination with AI feeds. Retrieved from <https://www.techinasia.com/bytedanceoverseas-expansion-strategy-break-down>.
- [17] Stefanko L. (2019, July 19). With FaceApp in the spotlight, new scams emerge <https://www.welivesecurity.com/2019/07/19/faceapp-spotlight-scams-emerge/>.

- [18] Aboshady A. (2019, July 14). The New “FaceApp” Trend: a Universal Trap or Just an Entertaining Program?. Retrieved from <https://identity-mag.com/the-new-faceapp-trend-just-for-fun-or-something-more/>.
- [19] Kelly C. (2019, July 28). Aging app FaceApp is giving people a warped idea of what getting older truly feels like. Retrieved from <https://www.nbcnews.com/think/opinion/aging-app-faceapp-giving-people-warped-idea-what-getting-older-ncna1034796>.
- [20] Protect Young Eyes (2018, August 2). What is the TikTok app?. Retrieved from <https://protectyouneyes.com/apps/tiktok-parental-controls/>.
- [21] Zhang K. (2018, May 19). I risked my life, please like! Mobile app Tik Tok has Hong Kong children craving acceptance – and some are going to dangerous extremes. Retrieved from <https://www.scmp.com/news/hong-kong/community/article/2146904/i-risked-my-life-please-mobile-app-tik-tok-has-hong-kong>.
- [22] Internet Matter (2019). How safe is TikTok app?. Retrieved from <https://www.internetmatters.org/hub/esafety-news/tik-tok-app-safety-what-parents-need-to-know/>.
- [23] Sturmer J. and Ockenden W. (2017, April 27). FaceApp: Experts have privacy concerns about popular face transformation app. Retrieved from <https://www.abc.net.au/news/2017-04-27/should-you-worry-about-privacy-when-using-faceapp/8476666>.
- [24] BBC News (2019, July 18). FaceApp: Chuck Schumer asks for FBI investigation. Retrieved from <https://www.bbc.com/news/world-us-canada-49027155>.
- [25] Gonzalez, H., Kadir, A. A., Stakhanova, N., Alzahrani, A. J., & Ghorbani, A. A. (2015, April). Exploring reverse engineering symptoms in Android apps. In Proceedings of the Eighth European Workshop on System Security (pp. 1-7).
- [26] Rastogi, S., Bhushan, K., & Gupta, B. B. (2016). Android applications repackaging detection techniques for smartphone devices. *Procedia Computer Science*, 78(C), 26-32.
- [27] Lim, K., Jeong, Y., Cho, S. J., Park, M., & Han, S. (2016). An Android Application Protection Scheme against Dynamic Reverse Engineering Attacks. *JoWUA*, 7(3), 40-52.
- [28] Sporea, I., Aziz, B., & McIntyre, Z. (2012). On the availability of anti-forensic tools for smartphones. *International Journal of Security*, 6(4), 58-64.
- [29] Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012, February). Detecting repackaged smartphone applications in third-party android marketplaces. In Proceedings of the second ACM conference on Data and Application Security and Privacy (pp. 317-326).
- [30] Balebako, R., Marsh, A., Lin, J., Hong, J. I., & Cranor, L. F. (2014). The privacy and security behaviors of smartphone app developers.
- [31] Krishnan, S., & Chen, L. (2019). Legal Concerns and Challenges in Cloud Computing. *arXiv preprint arXiv:1905.10868*.
- [32] Solanas A. and Mart´inez-Ballest´e A. (2009, August). Advances in Artificial Intelligence for Privacy Protection and Security. Retrieved from <https://www.worldscientific.com/worldscibooks/10.1142/6707>.
- [33] Desnos A., Gueguen G., Bachmann S. (2018). Introduction Androguard 3.4 Documentation. Retrieved from <https://androguard.readthedocs.io/en/latest/intro/>.

- [34] Download Ubuntu Desktop (2020). Retrieved from <https://ubuntu.com/download/desktop>.
- [35] Apkpure (2019). FaceApp for Android APK Download. Retrieved from <https://apkpure.com/faceapp/io.faceapp>.
- [36] Apkpure (2019). TikTok Download. Retrieved from <https://apkpure.com/tiktok-musically/com.zhiliaoapp.musically>.
- [37] FaceApp (2019, December 3). Privacy Policy. Retrieved from <https://www.faceapp.com/privacy-en.html>.
- [38] TikTok (2019, February). Privacy Policy. Retrieved from <https://www.tiktok.com/legal/privacy-policy-row?lang=en>.
- [39] TikTok (2020, January). Privacy Policy for Younger Users. Retrieved from <https://www.tiktok.com/legal/privacy-policy-for-younger-users?lang=en>.