FORENSICS AND eDISCOVERY

Sneha Sivaram

x23192054

Facebook App Investigation

Executive Summary

In this report, a simple analysis of the Facebook APK file has been carried out using some user-friendly and freely available tools and resources. I used Kali Linux Virtual Machine as a sandboxed environment where I could study the app's structure. During the analysis, the APK file was first decompiled using a tool called androguard which decompiled the apk files into java files which was readable. This tool was used to examine an important file of the app called androidmanifest.xml. The file contained many uses permissions and some of the main components that structured the Facebook application. Not only this file but after the decompiling process, an entire folder was generated which contained various files related to the app's components.

After the decompiling procedure, an online malware analysis tool was used to check for any malware present in the APK, there were some minor issues present. It is not sure whether they pose higher risks or if it is something harmless. Screenshots were taken to support the report.

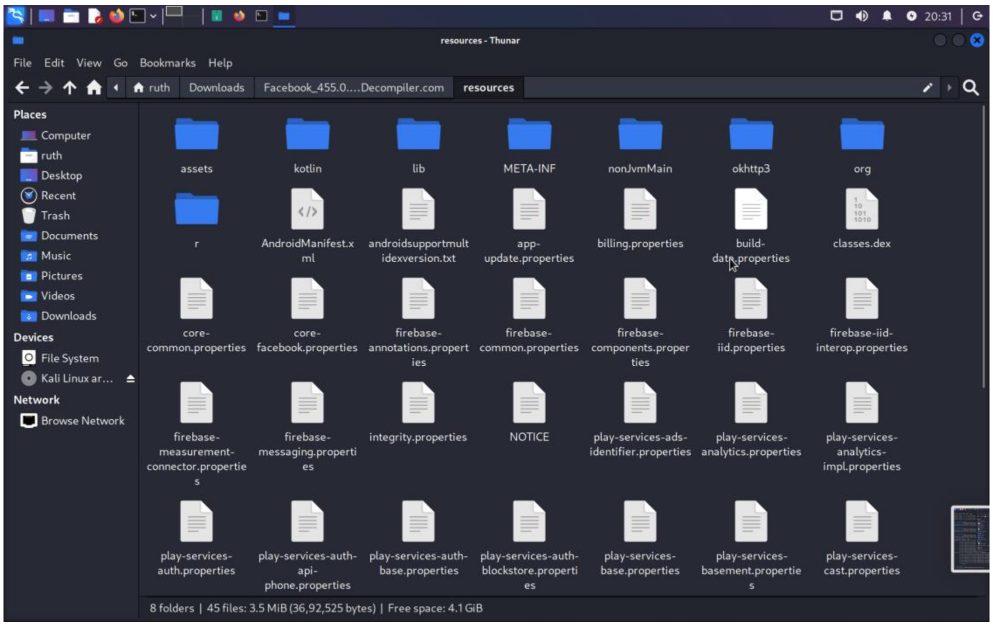
Finally, the Facebook app was imported to android studio and emulated using a virtual device. Using a feature present in the android studio, I was able to check for some logs whenever I interacted with the app. A test account was created for this very purpose of testing the app behaviour. The app seemed to be working fine without any issues, but when I logged out and tried logging in again I faced some difficulties, later I was able to login again. Some screenshot were taken as a support for my report. I used the android studio in my primary system Mac OS, since it the app would only work well in a primary system and I wasn't able to do the same in my virtual machine.

Methodology

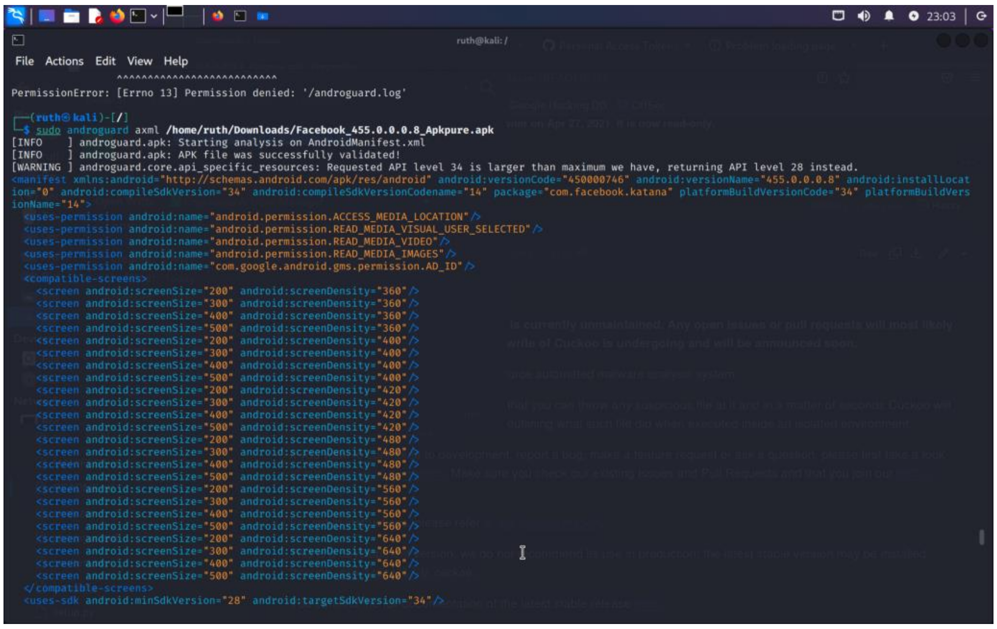
To start with my report, I searched for investigation reports related to my topic on google scholar which consists of many important conference papers and articles. I was able to find one which was closer to my given topic which was "Digital Forensic Analysis of Facebook App in Virtual Environment" [1]. According to this conference paper, the researchers have seemed to use a virtual device for testing the app behaviour especially an android device and an android emulator called 'Genymotion'. The reason that they chose this method is because a virtual device can be created again in case if the experimentation goes wrong but it is not the same with a physical device, since there are chances it might get damaged during the forensic experiment. Agrawal et al. [1].

I installed and used the tool called Androguard in Kali Linux VM to analyze the Facebook APK file, this tool is an android application analysis tool which is mainly used for proper examination of android application files [2]. It focuses on dissecting the Android Package File of the app, this procedure is also known as decompiling which allows the user to extract the source code and other resources of the app for deep understanding of the app structure [2]. This method is useful in understanding the app internally and also identify any possible vulnerabilities [2].

The tool then decompiled the apk into different files containing source codes, xml files, and some kind of images like the app logos and other icons.


resources - Thunar
File Edit View Go Bookmarks Help
ruth
Downloads
Facebook_455.0....Decompiler.com
resources
Places
Computer
ruth
Desktop
Recent
Trash
Documents
Music
Pictures
Videos
Downloads
Devices
File System
Kali Linux ar...
Network
Browse Network
assets
kotlin
lib
META-INF
nonJvmMain
okhttp3
org
r
AndroidManifest.xml
androidsupportmultidexversion.txt
app-update.properties
billing.properties
build-data.properties
classes.dex
core-common.properties
core-facebook.properties
firebase-annotations.properties
firebase-common.properties
firebase-components.properties
firebase-iid.properties
firebase-iid-interop.properties
firebase-measurement-connector.properties
firebase-messaging.properties
integrity.properties
NOTICE
play-services-ads-identifier.properties
play-services-analytics.properties
play-services-analytics-impl.properties
play-services-auth.properties
play-services-auth-api-phone.properties
play-services-auth-base.properties
play-services-auth-blockstore.properties
play-services-base.properties
play-services-basement.properties
play-services-cast.properties
8 folders | 45 files: 3.5 MiB (36,92,525 bytes) | Free space: 4.1 GiB

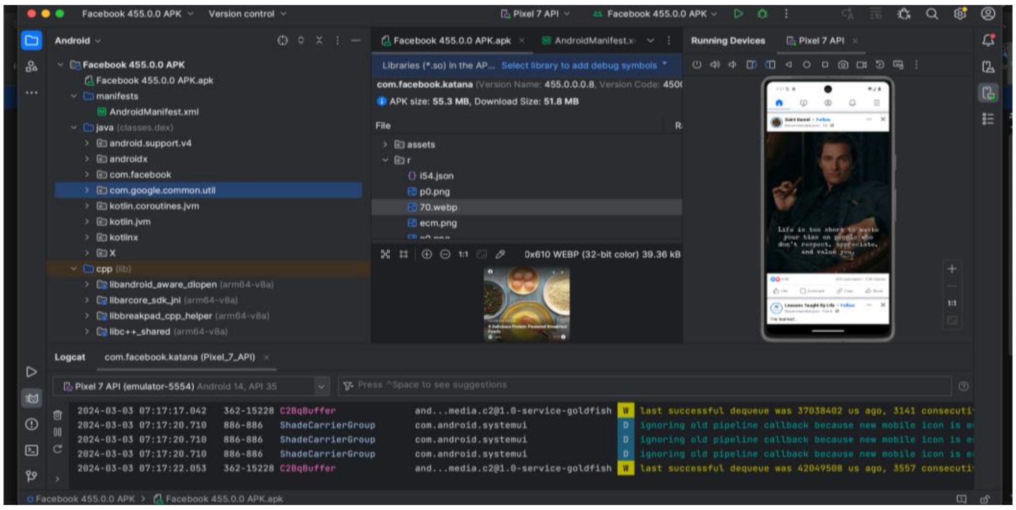
These were the files, I was able to access after decompilation.


```
PermissionError: [Errno 13] Permission denied: '/androguard.log'

┌──(ruth㉿kali)-[/]
└─$ sudo androguard axml /home/ruth/Downloads/Facebook_455.0.0.0.8_Apkpure.apk
[INFO    ] androguard.apk: Starting analysis on AndroidManifest.xml
[INFO    ] androguard.apk: APK file was successfully validated!
[WARNING ] androguard.core.api_specific_resources: Requested API level 34 is larger than maximum we have, returning API level 28 instead.
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="450000746" android:versionName="455.0.0.0.8" android:installLocat
ion="0" android:compileSdkVersion="34" android:compileSdkVersionCodename="14" package="com.facebook.katana" platformBuildVersionCode="34" platformBuildVers
ionName="14">
  <uses-permission android:name="android.permission.ACCESS_MEDIA_LOCATION"/>
  <uses-permission android:name="android.permission.READ_MEDIA_VISUAL_USER_SELECTED"/>
  <uses-permission android:name="android.permission.READ_MEDIA_VIDEO"/>
  <uses-permission android:name="android.permission.READ_MEDIA_IMAGES"/>
  <uses-permission android:name="com.google.android.gms.permission.AD_ID"/>
  <compatible-screens>
    <screen android:screenSize="200" android:screenDensity="360"/>
    <screen android:screenSize="300" android:screenDensity="360"/>
    <screen android:screenSize="400" android:screenDensity="360"/>
    <screen android:screenSize="500" android:screenDensity="360"/>
    <screen android:screenSize="200" android:screenDensity="400"/>
    <screen android:screenSize="300" android:screenDensity="400"/>
    <screen android:screenSize="400" android:screenDensity="400"/>
    <screen android:screenSize="500" android:screenDensity="400"/>
    <screen android:screenSize="200" android:screenDensity="420"/>
    <screen android:screenSize="300" android:screenDensity="420"/>
    <screen android:screenSize="400" android:screenDensity="420"/>
    <screen android:screenSize="500" android:screenDensity="420"/>
    <screen android:screenSize="200" android:screenDensity="480"/>
    <screen android:screenSize="300" android:screenDensity="480"/>
    <screen android:screenSize="400" android:screenDensity="480"/>
    <screen android:screenSize="500" android:screenDensity="480"/>
    <screen android:screenSize="200" android:screenDensity="560"/>
    <screen android:screenSize="300" android:screenDensity="560"/>
    <screen android:screenSize="400" android:screenDensity="560"/>
    <screen android:screenSize="500" android:screenDensity="560"/>
    <screen android:screenSize="200" android:screenDensity="640"/>
    <screen android:screenSize="300" android:screenDensity="640"/>
    <screen android:screenSize="400" android:screenDensity="640"/>
    <screen android:screenSize="500" android:screenDensity="640"/>
  </compatible-screens>
  <uses-sdk android:minSdkVersion="28" android:targetSdkVersion="34"/>
```

Then, I used the androguard tool to analyze the AndroidManifest.xml file listed above.

Justification: The decompilation feature of this tool makes it easier for the users to convert APK into easily readable java source code [2]. And its also easy to use.

Then, I used Android studio in my primary system Mac OS for emulating the Facebook app and check its behaviour. This is a great tool for building Android apps and it is mostly used by the developers [3]. Emulation is one of its main features and it also offers an option to select a virtual device to test the apps on it as well [3].
It has an AVD manager which helps in managing the virtual devices [3].
The Facebook APK was imported into this tool and then I dragged the file to the virtual device to directly install the app. This tool has a feature called logcat which would store the logs whenever the virtual device is being interacted with.



```
com.facebook.katana (Version Name: 455.0.0.0.8, Version Code: 450
APK size: 55.3 MB, Download Size: 51.8 MB
```

```
2024-03-03 07:17:17.042   362-15228 C2BqBuffer          and...media.c2@1.0-service-goldfish  W  last successful dequeue was 37038402 us ago, 3141 consecuti
2024-03-03 07:17:20.710   886-886   ShadeCarrierGroup   com.android.systemui                 D  ignoring old pipeline callback because new mobile icon is e
2024-03-03 07:17:20.710   886-886   ShadeCarrierGroup   com.android.systemui                 D  ignoring old pipeline callback because new mobile icon is e
2024-03-03 07:17:20.710   886-886   ShadeCarrierGroup   com.android.systemui                 D  ignoring old pipeline callback because new mobile icon is e
2024-03-03 07:17:22.053   362-15228 C2BqBuffer          and...media.c2@1.0-service-goldfish  W  last successful dequeue was 42049508 us ago, 3557 consecuti
```

I used Android Studio to emulate the app and check its behaviour.

Justification: Its emulator feature is easier to work with and it also supports wide range of devices as well to emulate an android application.

Test Environment Setup:

1. Operating System:

Host OS: Mac M1

- MacOS Sonoma 14.1
- Virtualization Software: VMware Fusion

Virtual Machine OS

- Kali Linux (latest version)
- Used a sandbox environment

2. Configuration:

- Mac OS with 8GB RAM with 8 cores and 256 GB storage space
- Kali Linux VM with 2GB RAM with 2 cores and 25 GB storage space

3. Precaution and Sandbox Measures

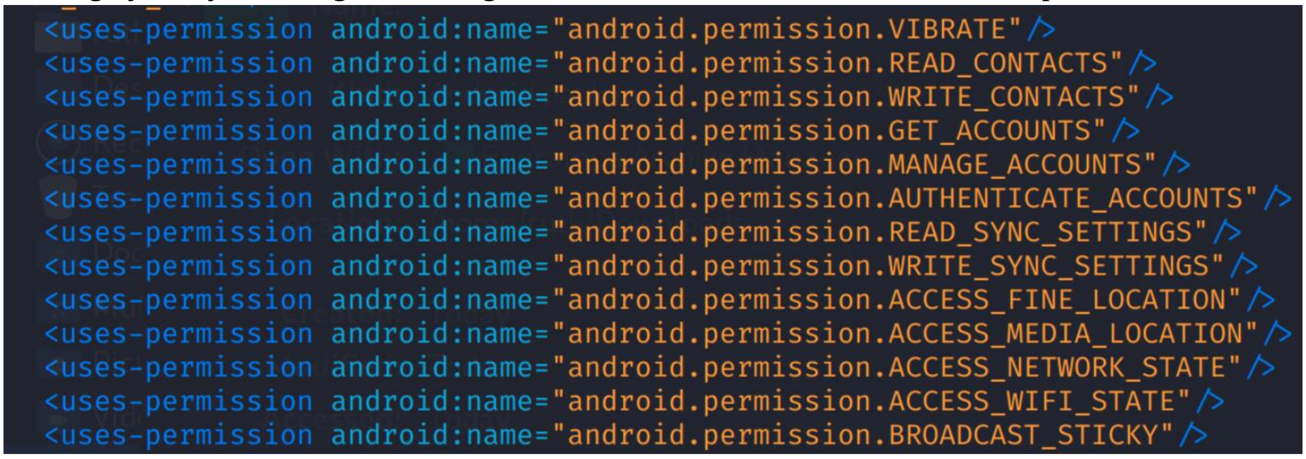
- Using virtualization software to build virtual machines and used it as a sandbox environment, so that it is isolated from the primary system.
- I took snapshots of my VM machine before analyzing the app in case if any problem occurs.

- The virtual machine’s network was isolated from the host system to ensure any problematic exposure.
- Firewalls was configured properly in both host and virtual systems.

App Investigation and Findings

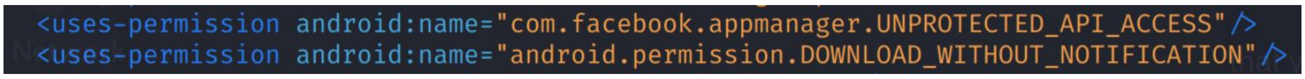
Sources of Forensic Artifacts

The androidmanifest.xml file contains valuable information about the app and it also describes the components of the application and declares permissions required by the app [4]. This file was thoroughly analyzed using the androguard tool and I was able to note some important information:



```
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS"/>
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS"/>
<uses-permission android:name="android.permission.READ_SYNC_SETTINGS"/>
<uses-permission android:name="android.permission.WRITE_SYNC_SETTINGS"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_MEDIA_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.BROADCAST_STICKY"/>
```

The app seemed to have a lot of permissions which may not be a right thing always as users may never remember always for what exactly their app has been permitted to do.



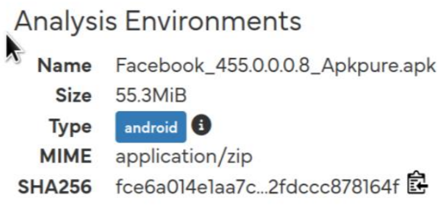
```
<uses-permission android:name="com.facebook.appmanager.UNPROTECTED_API_ACCESS"/>
<uses-permission android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION"/>
```

I noted down this particular line which says unprotected API access. Although it might not be a big problem but this permission could have security issues if it provides API access without protection.
Some of the permissions seems to be asking for information such as battery state and phone state which are sensitive data

.

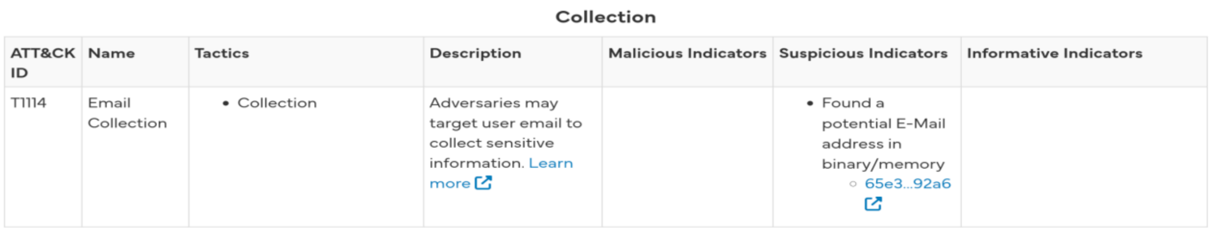
Malware Analysis of the APK

I have also tried using an online malware analysis tool to check for an issues within the APK file:



Analysis Environments

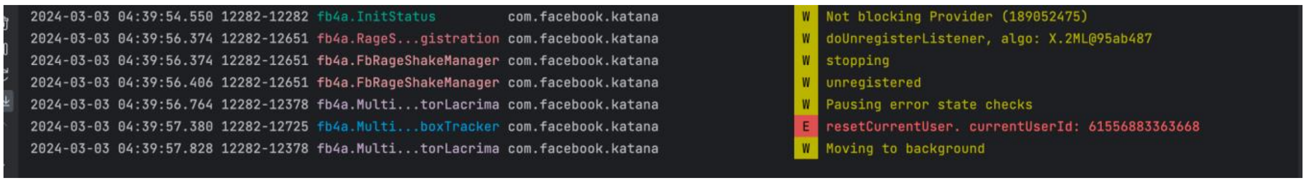
Name	Facebook_455.0.0.0.8_Apkpure.apk
Size	55.3MiB
Type	android
MIME	application/zip
SHA256	fce6a014e1aa7c...2fdccc878164f



Collection

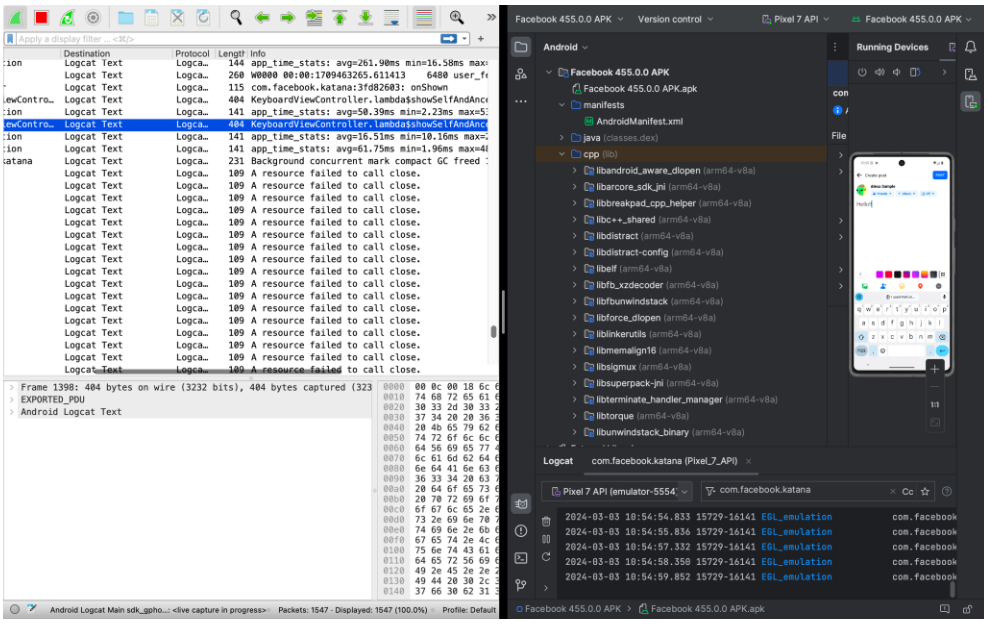
ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1114	Email Collection	• Collection	Adversaries may target user email to collect sensitive information. Learn more		• Found a potential E-Mail address in binary/memory ◦ 65e3...92a6	

Logs from android studio emulator:



```
2024-03-03 04:39:54.550 12282-12282 fb4a.InitStatus          com.facebook.katana   W  Not blocking Provider (189052475)
2024-03-03 04:39:56.374 12282-12651 fb4a.RageS...gistration com.facebook.katana   W  doUnregisterListener, algo: X.2ML@95ab487
2024-03-03 04:39:56.374 12282-12651 fb4a.FbRageShakeManager com.facebook.katana   W  stopping
2024-03-03 04:39:56.406 12282-12651 fb4a.FbRageShakeManager com.facebook.katana   W  unregistered
2024-03-03 04:39:56.764 12282-12378 fb4a.Multi...torLacrima com.facebook.katana   W  Pausing error state checks
2024-03-03 04:39:57.380 12282-12725 fb4a.Multi...boxTracker com.facebook.katana   E  resetCurrentUser. currentUserId: 61556883363668
2024-03-03 04:39:57.828 12282-12378 fb4a.Multi...torLacrima com.facebook.katana   W  Moving to background
```

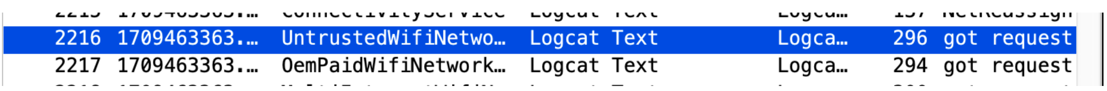
It recorded the logs whenever the app was closed and the device goes to background.
I have also tried using wireshark and captured the logs when using the app.



```
         Destination    Protocol  Length Info
:ion     Logcat Text    Logca…    144 app_time_stats: avg=261.90ms min=16.58ms max=
         Logcat Text    Logca…    260 W0000 00:00:1709463265.611413    6480 user_f
         Logcat Text    Logca…    115 com.facebook.katana:3fd82603: onShown
lewContro… Logcat Text  Logca…    404 KeyboardViewController.lambda$showSelfAndAnce
:ion     Logcat Text    Logca…    141 app_time_stats: avg=50.39ms min=2.23ms max=5
lewContro… Logcat Text  Logca…    404 KeyboardViewController.lambda$showSelfAndAnce
:ion     Logcat Text    Logca…    141 app_time_stats: avg=16.51ms min=10.16ms max=
:ion     Logcat Text    Logca…    141 app_time_stats: avg=61.75ms min=1.96ms max=4
katana   Logcat Text    Logca…    231 Background concurrent mark compact GC freed
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.
         Logcat Text    Logca…    109 A resource failed to call close.

> Frame 1398: 404 bytes on wire (3232 bits), 404 bytes captured (323
> EXPORTED_PDU
> Android Logcat Text

0000  00 0c 00 18 6c
0010  74 68 72 65 61
0020  30 33 2d 30 33
0030  37 34 20 20 36
0040  20 4b 65 79 62
0050  74 72 6f 6c 6c
0060  64 56 69 65 77
0070  6c 61 6d 62 64
0080  6e 64 41 6e 63
0090  36 33 34 20 63
00a0  20 64 6f 65 73
00b0  20 70 72 69 6f
00c0  6f 67 6c 65 2e
00d0  73 2e 69 6e 70
00e0  74 69 6e 2e 6b
00f0  67 65 74 2e 4c
0100  75 6e 74 43 61
0110  64 65 72 56 69
0120  49 2e 45 2e 2e
0130  49 44 20 30 2c
0140  37 66 30 62 31

Android Logcat Main sdk_gpho...: <live capture in progress>   Packets: 1547 · Displayed: 1547 (100.0%)   Profile: Default
```

```
Logcat   com.facebook.katana (Pixel_7_API)
Pixel 7 API (emulator-5554)   com.facebook.katana
2024-03-03 10:54:54.833 15729-16141 EGL_emulation            com.facebook
2024-03-03 10:54:55.836 15729-16141 EGL_emulation            com.facebook
2024-03-03 10:54:57.332 15729-16141 EGL_emulation            com.facebook
2024-03-03 10:54:58.350 15729-16141 EGL_emulation            com.facebook
2024-03-03 10:54:59.852 15729-16141 EGL_emulation            com.facebook
```

Facebook 455.0.0 APK
Version control
Pixel 7 API
Android
Running Devices
Facebook 455.0.0 APK.apk
manifests
AndroidManifest.xml
java (classes.dex)
cpp (lib)
libandroid_aware_dlopen (arm64-v8a)
libbarcore_sdk_jni (arm64-v8a)
libbreakpad_cpp_helper (arm64-v8a)
libc++_shared (arm64-v8a)
libdistract (arm64-v8a)
libdistract-config (arm64-v8a)
libelf (arm64-v8a)
libfb_xzdecoder (arm64-v8a)
libfbunwindstack (arm64-v8a)
libforce_dlopen (arm64-v8a)
liblinkerutils (arm64-v8a)
libmemalign16 (arm64-v8a)
libsigmux (arm64-v8a)
libsuperpack-jni (arm64-v8a)
libterminate_handler_manager (arm64-v8a)
libtorque (arm64-v8a)
libunwindstack_binary (arm64-v8a)



```
2216  1709463363.…  UntrustedWifiNetwo…  Logcat Text         Logca…   296  got request
2217  1709463363.…  OemPaidWifiNetwork…  Logcat Text         Logca…   294  got request
```

Conclusion

In this simple forensic investigation of the facebook app, some of the important information was collected using tools like androguard to analyze the apk and generate the java source code for easy understanding and deeply analyzing the android manifest file to understand the app structure and its permissions. Then using android studio emulator to see the app behaviour and capture the logs. And checking the security of the apk tool using online malware analysis tools. Although, I was able to find some important data by using these tools, I had limited options because of using MacOS M1 as the host system. The tool compatibility is quite lower and limited when compared to a windows OS. And, I believe that more advanced tools could have been used for an in-depth detailed analysis. Advanced tools in kali machine such as Ghidra and Radare2 can be used for better analysis of the apk file. Also, emulating the app on a virtual device doesn't perform to its full potential than in an actual device so there are limitations to that as well. Because it would not capture real world scenarios. Additionally, it also better to have dedicated virtual environment such as a cuckoo sandbox in kali linux which would have been useful in capturing the proper app behaviour. In conclusion, it is important to update the forensics investigation methods in order to be able to investigate the applications well and also with proper security measures.

REFERENCES

[1] A. K. Agrawal, A. Sharma and P. Khatri, “Digital Forensic Analysis of Facebook App in Virtual Environment,” *in 2019 6th Int. Conf. on Computing for Sustainable Global Development (INDIACom), New Delhi, India, Mar. 13-15, 2019,* pp. 660-664. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8991287
[Accessed on: Feb. 29, 2024].

[2] S. Mishra, “Androguard: A comprehensive tool for in-depth Android Security Analysis,” *Medium,* Sept. 15, 2023. [Online]. Available: https://medium.com/@careertechnologymiraroad/androguard-a-comprehensive-tool-for-in-depth-android-security-analysis-adeb334d9c6c [Accessed on: Mar. 1, 2024]

[3] A. Niaz, “Android Studio And its Features,” *Tech Blogs,* June 6, 2022. [Online]. Available: https://techblogs.42gears.com/android-studio-and-its-features/ [Accessed on: Mar. 1, 2024]

[4] B. Sharma, “Androidmanifest.xml,” *Medium,* Dec, 10, 2019. [Online]. Available: https://medium.com/android-news/androidmanifest-xml-4d5a3e821f91 [Accessed on: Mar. 1, 2024]