Android Forensic Analysis Using Open-Source Tools

Final Report

Umesh Kalupathirannehelage – 300389749 (Team Lead)

Sahan Pattinikuttige Nonis – 300389470

Group 7 (F25_4440_G7)

Mobile Cybersecurity

CSIS4440 - 003

Archival Video Link - https://drive.google.com/file/d/1wXcxxIeNzRZm8vYZDD3WPdheJwGbz_FV/view?usp=drive_link

Abstract - Umesh

This project used ADB to extract Dropbox and Instagram Lite data, followed by full /data/data analysis with ALEAPP to recover timestamps, cached media, and app artifacts. Pinterest was examined using MobSF static analysis to evaluate permissions, trackers, and security posture. Combined, these tools provided detailed forensic insight across app behaviors and stored evidence.

Abstract – Sahan

This report presents mobile security analysis using forensic tools Drozer and JADX across consumer and security-critical Android applications. Spotify was used to map real-world IPC exposure, while Trust Wallet was examined for secure component posture. InsecureBankv2 demonstrates exported component abuse via activity launch bypass. Findings highlight dynamic vs static analysis utility and security contrasts.

Table of Contents

1. Background Research....4
   1.1 Application Analysis: Dropbox....4
   1.2 Application Analysis: Instagram Lite....4
   1.3 Tool Analysis: ALEAPP....4
   1.4 Tool Analysis: MobSF....4
   1.5 Strava....4
   1.6 Drozer & JADX....5
2. Forensic Plan....5
   2.1 Dropbox....5
   2.2 Instagram Lite....5
   2.3 ALEAPP....5
   2.4 MobSF....5
   2.5 Strava....5
   2.6 Drozer & JADX....6
      How the tools work....6
      How the examination will be conducted....6
      Underlying architecture / forensic layer insight....6
3. Detailed Analysis....6
   3.1 Dropbox....6
   3.2 Instagram Lite....12
   3.3 ALEAPP....13
      3.3.1 ULR User Preferences....13
      3.3.2 Bluetooth Adapter Information....14
      3.3.3 Calendar (calendar.db)....15
      3.3.4 Call Logs (calllog.db)....15
      3.3.5 Google Chrome Data....16
      3.3.4 Contacts....20
      3.3.4 Sim Information....21
      3.3.5 Settings - Secure....22
      3.3.5 Digital Wellbeing (App Usage Events)....22
      3.3.6 Emulated Storage....23
      3.3.7 Google Maps – Labeled Places....24

3.3.8 Gmail Active Account 25
3.3.9 Google Photos Cache 25
3.3.10 Google Photos Local Media 26
3.3.11 Installed Applications 27
3.3.12 Last Boot Time 27
3.3.13 SMS Messages 27
3.3.14 Wi-Fi Hotspot Configuration 28
3.4 MobSF 29
3.5 Strava 35
3.6 Drozer 37
3.7 Spotify Analysis using Drozer 37
3.8 Trust Wallet Analysis using Drozer 39
3.9 InsecureBankv2 (Vulnerable App) Analysis using Drozer 40
3.10 JADX 42
3.11 Spotify Analysis using JADX 43
3.12 Trust Wallet Analysis using JADX 44
3.13 Integrated Component Prototype 47
4. Results and Insights 49
4.1 ALEAPP 49
4.2 MobSF (Static Analysis) 49
4.3 Dropbox 50
4.4 Instagram Lite 50
4.5 Strava 51
4.6 Drozer, JADX, Spotify, TrustWallet & InsecureBankv2 51
How was the exploration useful? 52
Limits of research / exploration 52
5. Conclusions 52
5.1 Tools & Applications - Umesh 52
5.2 Tools & Applications - Sahan 53
Future Work 53
Other scenarios where these insights apply 53
Broad conclusion 53
6. AI Use Section 53
7. Work Log Table 54
8. Appendix 59
9. References 60

1. Background Research

1.1 Application Analysis: Dropbox

Dropbox is a cloud-storage application that synchronizes files across devices. Unlike purely cloud-based applications, Dropbox stores significant local metadata, such as cached thumbnails, WebView content, notification data, and user interaction logs. In Android forensic investigations, Dropbox is a valuable target because it generates persistent artifacts under /data/data/com.dropbox.android. These artifacts remain even when files are deleted from the cloud, making the app suitable for reconstruction of user actions. Since sensitive apps such as WhatsApp, Maps, Gmail, and Chrome were excluded from this project, Dropbox served as a practical non-restricted app capable of producing structured forensic evidence.

1.2 Application Analysis: Instagram Lite

Instagram Lite is a lightweight version of Instagram designed for low-resource devices. Instead of downloading large databases, it stores mostly temporary cached media, thumbnails, and partial WebView content. Although it does not generate SQLite-based message or post databases, it still creates valuable artifacts under /cache and /files, which can reveal user interactions. Because Strava failed repeatedly and could not authenticate on the emulator, Instagram Lite replaced it as the second application for analysis.

1.3 Tool Analysis: ALEAPP

ALEAPP (Android Logs Events And Protobuf Parser) is an open-source forensic triage tool designed to parse Android application artifacts and system logs. It supports over 300 artifact modules and can analyze both ADB extractions and full filesystem dumps. ALEAPP also provides visual HTML reports and timeline views, making it valuable for rapid triage. Since the project required exploration of novel forensic tools, ALEAPP was selected to demonstrate automated extraction and correlation of mobile artifacts.

1.4 Tool Analysis: MobSF

MobSF (Mobile Security Framework) is a widely used mobile analysis platform supporting both static analysis of APKs and dynamic behavior monitoring. Although originally designed for security analysis, Mobile Forensics can benefit from MobSF's insights into permissions, API calls, file system behavior, and dynamic logs. For this project, MobSF was used to evaluate Dropbox and Instagram Lite from a tooling perspective.

1.5 Strava

Strava is a fitness-tracking application that stores user activities, routes, GPS traces, performance metrics, and sync events. Typical Android artifacts include SQLite databases (e.g., athlete.db, activities.db), Mapbox/Strava FIT and GPX files, cached images, session tokens, and protobuf-based activity metadata. Prior research indicates that Strava heavily relies on native libraries for rendering maps and processing sensor/GPS data, which complicates extraction in emulator environments. Additionally, many artifacts are generated only after valid activities are successfully recorded or synced to the device.

1.6 Drozer & JADX

Drozer performs **dynamic IPC attack surface mapping**, simulating what another app could access on the same device. JADX performs **static decompilation** to expose app architecture and potential backend endpoints through string and manifest inspection.

2. Forensic Plan

2.1 Dropbox

The plan involved installing Dropbox on an Android Studio emulator, signing in, browsing folders, previewing images, viewing documents, and generating natural user activity. This usage causes creation of thumbnail caches (/cache/u*/thumbs), account metadata (app_DropboxSyncCache), contact data, WebView rendering caches, and SQLite databases. After activity generation, the entire /data/data/com.dropbox.android directory and relevant system data were extracted using ADB. The extraction was later merged into the master dataset for processing with ALEAPP.

2.2 Instagram Lite

The plan involved installing Instagram Lite, logging in, scrolling the feed, opening images, viewing profiles, liking posts, and browsing hashtags. These interactions generate cached images and media-processing files. The full /data/data/com.instagram.lite folder was then extracted using ADB and merged with the master dataset. ALEAPP was later used to parse the app directory to identify thumbnails, cached posts, and temporal artifacts.

2.3 ALEAPP

The plan was to combine Dropbox and Instagram Lite extractions with partial system directories and run ALEAPP once in filesystem mode (-t fs). The merged dataset included /data/data, /data/system, /data/misc, and /sdcard artifacts. This allowed ALEAPP to generate a multi-application, multi-directory analysis in a single report. The HTML, TSV, and timeline outputs were collected for reporting.

2.4 MobSF

The plan consisted of uploading the APK files into MobSF for static analysis and running the dynamic analyzer using the emulator. This allowed the extraction of API usage, permission structures, network connections, and file operations performed by the apps. The dynamic log traces were also used to correlate with artifacts found in ALEAPP.

2.5 Strava

My initial plan was:

1. Install Strava on the Android emulator.
2. Create a test account and generate a mock GPX route.
3. Import or sync the GPX to force Strava to create local artifacts.
4. Extract /data/data/com.strava/ using ADB root + pull.

5. Parse databases and file systems using:
    - DB Browser for SQLite
    - ALEAPP for timeline and cache extraction
    - Manual review of files, cache, and databases folders
6. Correlate Strava activity timestamps with Dropbox activity in the integrated dashboard.

This would have produced GPS trace artifacts, sync events, and cached map tiles required for timeline reconstruction.

2.6 Drozer & JADX

How the tools work

- Drozer performs dynamic security analysis through Android's IPC layer by interacting with exported components exactly like a malicious third-party app would, without root.
- JADX performs static APK reverse engineering, decompiling Dalvik bytecode into Java-like source and allowing analysts to inspect the manifest, package architecture, permissions, and embedded strings such as backend URLs.

How the examination will be conducted

Examination flow:

1. Connect to Drozer agent → enumerate device/apps
2. Map Spotify attack surface via Drozer → query providers
3. Map Trust Wallet commit exposure via Drozer
4. Demonstrate InsecureBankv2 component abuse (auth bypass)
5. Open Spotify & Trust Wallet APKs in JADX → inspect manifest + string search
6. Capture relevant screenshots for all working/failing behaviors

Underlying architecture / forensic layer insight

Drozer explores what the app exposes at runtime (Activities, Providers, Broadcasts, Services) while JADX reveals how those components are declared statically in the manifest and how the compiled code stores communication surfaces. The architecture contrast helps model Android sandbox posture for each app type (consumer vs crypto vs vulnerable training build).

3. Detailed Analysis

3.1 Dropbox

Analysis of the Dropbox application on the emulator focused on the set of SQLite databases under /data/data/com.dropbox.android/.

```
Command Prompt

C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj>
C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj>adb install apk\Dropbox.apk
Performing Streamed Install
Success

C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj>adb root
restarting adbd as root

C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj>adb shell
generic_x86_64_arm64:/ #
generic_x86_64_arm64:/ # pm list packages | grep dropbox
package:com.dropbox.android
generic_x86_64_arm64:/ # exit

C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj>adb pull /data/data/com.dropbox.android dropbox_data
adb: error: cannot create 'dropbox_data\files\frc_1:735665981150:android:4be547085a889c09_fireperf_activate.json': No such file or directory
```

```
C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj>tree dropbox_data
Folder PATH listing for volume Windows
Volume serial number is 92E1-2FA2
C:\USERS\UMESH KARUNARATHNA\ONEDRIVE\DESKTOP\MCS FINAL PROJ\DROPBOX_DATA
├───app_DropboxSyncCache
│   └───hizqkiq3952astb
│       ├───2765813185-account
│       │   ├───account_photo_cache
│       │   └───contact_cache
│       ├───2765813185-dbapp
│       ├───2765813185-notifications
│       │   └───files
│       ├───local-dbapp_noauth
│       └───Temp
├───app_textures
├───app_webview
│   └───Default
├───databases
├───files
├───global
│   └───external
│       ├───featuregating
│       └───miscthumbs
└───u2765813185
    ├───assistantrecents
    ├───featuregating
    ├───fileviewlogger
    ├───pendingcomments
    └───thumbs
        └───Photos
            ├───ChatGPT Image Aug 2, 2025, 10_55_45 PM.png
            └───Host vs Network IPS Comparison.png
```

The main content database (db.db) contains several tables that reconstruct the user's cloud folder tree and the files available offline. The dropbox table lists 30+ paths with metadata such as hash, icon, is_dir, mime_type, and parent_path. From this table it is possible to rebuild the logical folder structure used by the account, including top-level folders such as /School, /Work, /Projects, /Music, /Medical, /Documents, /Family and /Photos. Each entry includes the file type and canonical path, showing a mixture of PowerPoint slides (e.g., *VPN Demonstration Seminar.pptx*), Excel workbooks (e.g., *Gamified v Traditional Feedback.xlsx* and *medical_records.xlsx*), PDF reports (e.g., *regex.pdf*, *invoice.pdf* and *victory.mp3* receipts), Windows executables, MP3 files, and several cybersecurity-related images (for example *cyber-security-logo-with-…*, *phishing_attacks_1200x600.png* and *cybersecurity.png*). This demonstrates that the Dropbox account is being used to store both course material for CSIS4440 and other modules, as well as personal documents such as medical records and work receipts.

DB Browser for SQLite - C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\dropbox_data\databases\2765813185-db.db

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Undo | Open Project | Save Project | Attach Database | Close Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: dropbox | Filter in any column

	hash	icon	is_dir	path	canon_path	mime_type	thumb_exists	parent_path
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	3fc8f1	folder	1	/School	/school	NULL	0	/
2	3fc8f1	page_white	0	/School/VPN Demonstration Seminar.pkt	/school/vpn demonstration seminar.pkt	application/octet-stream	0	/School/
3	3fc8f1	page_white_excel	0	/School/Gamified v Traditional …	/school/gamified v traditional …	application/vnd.openxmlformats-…	1	/School/
4	3fc8f1	page_white_film	0	/Videos/SPEAR.mp4	/videos/spear.mp4	video/mp4	1	/Videos/
5	3fc8f1	page_white_picture	0	/Photos/Host vs Network IPS …	/photos/host vs network ips …	image/png	1	/Photos/
6	3fc8f1	page_white_picture	0	/Photos/socialmedia phishing2.webp	/photos/socialmedia phishing2.webp	image/webp	1	/Photos/
7	3fc8f1	page_white_picture	0	/Photos/Phishing_Attacks_1200x600.png	/photos/phishing_attacks_1200x600.png	image/png	1	/Photos/
8	3fc8f1	page_white_picture	0	/Photos/cyber-security-logo-with-…	/photos/cyber-security-logo-with-…	image/webp	1	/Photos/
9	3fc8f1	page_white_picture	0	/Photos/cyber-security-icon-shield-…	/photos/cyber-security-icon-shield-…	image/jpeg	1	/Photos/
10	3fc8f1	page_white_picture	0	/Photos/ChatGPT Image Aug 2, 2025, …	/photos/chatgpt image aug 2, 2025, …	image/png	1	/Photos/
11	3fc8f1	page_white_picture	0	/Photos/cyber-security-concept-data-…	/photos/cyber-security-concept-data-…	image/jpeg	1	/Photos/
12	3fc8f1	page_white_acrobat	0	/Documents/Regex.pdf	/documents/regex.pdf	application/pdf	1	/Documents/
13	3fc8f1	page_white_sound	0	/Music/background.mp3	/music/background.mp3	audio/mpeg	0	/Music/
14	3fc8f1	page_white_text	0	/Work/receipt-…	/work/receipt-…	text/plain	0	/Work/
15	3fc8f1	page_white_sound	0	/Music/victory.mp3	/music/victory.mp3	audio/mpeg	0	/Music/
16	3fc8f1	page_white_acrobat	0	/Documents/document.pdf	/documents/document.pdf	application/pdf	1	/Documents/
17	3fc8f1	page_white_excel	0	/Medical/Medical_Records.xlsx	/medical/medical_records.xlsx	application/vnd.openxmlformats-…	1	/Medical/
18	3fc8f1	page_white_sound	0	/Music/grizzly_roar.mp3	/music/grizzly_roar.mp3	audio/mpeg	0	/Music/
19	3fc8f1	page_white	0	/Projects/DB.Browser.for.SQLite-…	/projects/db.browser.for.sqlite-…	application/x-msi	0	/Projects/
20	3fc8f1	page_white	0	/Projects/putty-64bit-0.83-…	/projects/putty-64bit-0.83-…	application/x-msi	0	/Projects/
21	3fc8f1	page_white_gear	0	/Projects/nmap-7.98-setup.exe	/projects/nmap-7.98-setup.exe	application/x-msdos-program	0	/Projects/
22	3fc8f1	page_white_acrobat	0	/Documents/invoice.pdf	/documents/invoice.pdf	application/pdf	1	/Documents/
23	3fc8f1	page_white_acrobat	0	/Documents/Lec6_Slides.pdf	/documents/lec6_slides.pdf	application/pdf	1	/Documents/
24	3fc8f1	page_white_text	0	/Work/receipt-…	/work/receipt-…	text/plain	0	/Work/
25	3fc8f1	page_white_text	0	/Work/receipt-…	/work/receipt-…	text/plain	0	/Work/
26	3fc8f1	page_white_text	0	/Work/receipt-…	/work/receipt-…	text/plain	0	/Work/
27	3fc8f1	page_white_text	0	/Work/receipt-…	/work/receipt-…	text/plain	0	/Work/
28	3fc8f1	page_white_text	0	/Work/receipt-…	/work/receipt-…	text/plain	0	/Work/
29	3fc8f1	folder	1	/Documents	/documents	NULL	0	/
30	3fc8f1	folder	1	/Family	/family	NULL	0	/
31	3fc8f1	folder	1	/Medical	/medical	NULL	0	/

1 - 31 of 48 | Go to: 1

The photos table inside the same database focuses on media items that Dropbox treats as photos or videos. It records a canonical path, a time_taken timestamp, and flags such as shared_folder_status. Entries show a short spear-phishing video at /videos/spear.mp4 and multiple project-related images under /csis4440_project/, including screenshots used in the spear-phishing training (for example *13_p-600x337.png* and *cybersecurity.png*). The associated timestamps (stored as Unix milliseconds) align with the project activity window, so an examiner can correlate what material was uploaded around the time the user was preparing their final project. The thumbnail_info table complements this by recording which of these files have locally cached thumbnails, along with thumbnail resolutions such as 640x480_bestfit or 256x256_fit_one_bestfit. Even if the main content were removed later, the presence of thumbnail metadata for paths like /documents/document.pdf, /medical/medical_records.xlsx, /photos/host vs network ips … and /photos/socialmedia phishing2.webp would still reveal which documents and images existed in the user's Dropbox.

DB Browser for SQLite - C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\dropbox_data\databases\2765813185-db.db

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Undo | Open Project | Save Project | Attach Database | Close Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: photos | Filter in any column

	_id	item_id	sort_key	canon_path	time_taken	shared_folder_status	date_hidden_utc_ms
1	1	WCWiYh9PYuoAAAAAAAAAGg	V--0a_SAGDxzzzzzzzzzk----…	/videos/spear.mp4	1762401142000	1	NULL
2	2	WCWiYh9PYuoAAAAAAAAAGA	V--0a_S9KeSzzzzzzzzzzk----BEbwYlPO4Z…	/photos/host vs network ips …	1762401081000	1	NULL
3	3	WCWiYh9PYuoAAAAAAAAAFg	V--0a_S9IhSzzzzzzzzzzk----BEbwYlPO4Z…	/photos/socialmedia phishing2.webp	1762401079000	1	NULL
4	4	WCWiYh9PYuoAAAAAAAAAFQ	V--0a_S9IhSzzzzzzzzzzk----BEbwYlPO4Z…	/photos/phishing_attacks_1200x600.png	1762401079000	1	NULL
5	5	WCWiYh9PYuoAAAAAAAAAFA	V--0a_S9IhSzzzzzzzzzzk----BEbwYlPO4Z…	/photos/cyber-security-logo-with-…	1762401079000	1	NULL
6	6	WCWiYh9PYuoAAAAAAAAAEw	V--0a_S9IhSzzzzzzzzzzk----BEbwYlPO4Z…	/photos/cyber-security-icon-shield-…	1762401079000	1	NULL
7	7	WCWiYh9PYuoAAAAAAAAAEg	V--0a_S9IhSzzzzzzzzzzk----BEbwYlPO4Z…	/photos/chatgpt image aug 2, 2025, …	1762401079000	1	NULL
8	8	WCWiYh9PYuoAAAAAAAAAEQ	V--0a_S9IhSzzzzzzzzzzk----BEbwYlPO4Z…	/photos/1613030086908.png	1762401079000	1	NULL
9	9	WCWiYh9PYuoAAAAAAAAAEA	V--0a_S9Hixzzzzzzzzzzk----BEbwYlPO4Z…	/photos/cyber-security-concept-data-…	1762401078000	1	NULL
10	10	WCWiYh9PYuoAAAAAAAAAMQ	V--0afQ4T-xzzzzzzzzzk----…	/csis4440_project/…	1764002986000	1	NULL
11	11	WCWiYh9PYuoAAAAAAAAAMg	V--0afQ4VwSzzzzzzzzzk----…	/csis4440_project/…	1764002989000	1	NULL
12	12	WCWiYh9PYuoAAAAAAAAAMA	V--0afWzYxSzzzzzzzzzk----…	/csis4440_project/13_p-600x337.png	1764031959000	1	NULL
13	13	WCWiYh9PYuoAAAAAAAAAMw	V--0afX-0FSzzzzzzzzzk----…	/csis4440_project/cybersecurity.png	1764031989000	1	NULL

User activity within Dropbox can be further reconstructed using the recents_entries and starred_info tables. recents_entries logs action_date, event_type, source_type (in this case typically antenna, indicating server-side events synced to the device) and a JSON file_path field. The entries show that, shortly before acquisition, the user accessed items across several categories: course documents in /School/, project artefacts in /Projects/, and multiple receipt files in /Work/receipts-…, as well as photos in the /Family/ and /Music/ folders.

DB Browser for SQLite - C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\dropbox_data\databases\2765813185-db.db

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Undo | Open Project | Save Project | Attach Database | Close Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: recents_entries | Filter in any column

	_id	action_date	batch_id	event_type	object_type	sort_priority	source_type	update_date	file_path	shared_link_name	shared_link_thur
1	1	1763534411000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	9	0	1	antenna	1764031715567	{"Path":"\/Documents\/…	NULL	NULL
2	2	1763534411000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	9	0	2	antenna	1764031715567	{"Path":"\/Videos\/…	NULL	NULL
3	3	1763534411000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	9	0	3	antenna	1764031715567	{"Path":"\/Music\/…	NULL	NULL
4	4	1763534411000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	9	0	4	antenna	1764031715567	{"Path":"\/Work\/receipt-…	NULL	NULL
5	5	1763534411000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	9	0	5	antenna	1764031715567	{"Path":"\/Music\/…	NULL	NULL
6	6	1762839085000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	9	0	6	antenna	1764031715567	{"Path":"\/Documents\/…	NULL	NULL
7	7	1762403532000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	7	antenna	1764031715567	{"Path":"\/Medical\/…	NULL	NULL
8	8	1762403169000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	8	antenna	1764031715567	{"Path":"\/Music\/…	NULL	NULL
9	9	1762403136000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	9	antenna	1764031715567	{"Path":"\/School\/Gamified v …	NULL	NULL
10	10	1762403135000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	10	antenna	1764031715567	{"Path":"\/School\/VPN Demonstration…	NULL	NULL
11	11	1762403124000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	11	antenna	1764031715567	{"Path":"\/Projects\/…	NULL	NULL
12	12	1762403123000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	12	antenna	1764031715567	{"Path":"\/Projects\/…	NULL	NULL
13	13	1762403122000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	13	antenna	1764031665732	{"Path":"\/Projects\/nmap-7.98-…	NULL	NULL
14	14	1762403050000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	14	antenna	1764031665732	{"Path":"\/Documents\/…	NULL	NULL
15	15	1762403047000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	15	antenna	1764031665732	{"Path":"\/Documents\/…	NULL	NULL
16	16	1762401238000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	16	antenna	1764031665732	{"Path":"\/Work\/receipt-…	NULL	NULL
17	17	1762401238000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	17	antenna	1764031665732	{"Path":"\/Work\/receipt-…	NULL	NULL
18	18	1762401238000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	18	antenna	1764031665732	{"Path":"\/Work\/receipt-…	NULL	NULL
19	19	1762401238000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	19	antenna	1764031665732	{"Path":"\/Work\/receipt-…	NULL	NULL
20	20	1762401237000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	20	antenna	1764031665732	{"Path":"\/Work\/receipt-…	NULL	NULL
21	21	1761767848000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	21	antenna	1764031665732	{"Path":"\/Documents","IsDir":true}	NULL	NULL
22	22	1761767848000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	22	antenna	1764031665732	{"Path":"\/Family","IsDir":true}	NULL	NULL
23	23	1761767848000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	23	antenna	1764031665732	{"Path":"\/Medical","IsDir":true}	NULL	NULL
24	24	1761767848000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	24	antenna	1764031665732	{"Path":"\/Music","IsDir":true}	NULL	NULL
25	25	1761767848000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	25	antenna	1764031665732	{"Path":"\/Personal","IsDir":true}	NULL	NULL
26	26	1761767848000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	26	antenna	1764031665732	{"Path":"\/Photos","IsDir":true}	NULL	NULL
27	27	1761767848000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	27	antenna	1764031665732	{"Path":"\/Projects","IsDir":true}	NULL	NULL
28	28	1761767848000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	28	antenna	1764031665732	{"Path":"\/School","IsDir":true}	NULL	NULL
29	29	1761767848000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	29	antenna	1764031665732	{"Path":"\/Videos","IsDir":true}	NULL	NULL
30	30	1761767848000	RmlsZUFjdGl2aXR5OldDV21ZaD1QWXVvQUFB…	0	0	30	antenna	1764031665732	{"Path":"\/Work","IsDir":true}	NULL	NULL
31	31	1764032231000	local-9-2025-10-25-0-/photos	9	0	-1	local_antenna	1764032231000	{"Path":"\/photos\/chatgpt image aug…	NULL	NULL

1 - 31 of 31 | Go to: 1

The starred_info table highlights items the user explicitly marked as important: for example,/csis4440_project/textfile.txt,/csis4440_project/cybersecurity.png,documents/regex.pdf, the *Gamified v Traditional Feedback.xlsx* spreadsheet, and the top-level /work folder are all stored as starred_state = SYNCED. This provides a clear indication of which project artefacts and personal documents the user prioritised.

DB Browser for SQLite - C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\dropbox_data\databases\2765813185-db.db

Table: starred_info

	unique_id	path	starred_id_type	starred_id	file_type	title	url	modified_time	starred_state
1	/csis4440_project/testfile.txt-FILE	/csis4440_project/testfile.txt	ENCODED_FILE_OBJ_ID	id:WCWiYh9PYuoAAAAAAAAAANA	FILE	testfile.txt		1764032042000	SYNCED
2	/csis4440_project/cybersecurity.png-…	/csis4440_project/cybersecurity.png	ENCODED_FILE_OBJ_ID	id:WCWiYh9PYuoAAAAAAAAAAMw	FILE	cybersecurity.png		1764031989000	SYNCED
3	/documents/regex.pdf-FILE	/documents/regex.pdf	ENCODED_FILE_OBJ_ID	id:WCWiYh9PYuoAAAAAAAAAAIw	FILE	Regex.pdf		1762403048000	SYNCED
4	/school/gamified v traditional …	/school/gamified v traditional …	ENCODED_FILE_OBJ_ID	id:WCWiYh9PYuoAAAAAAAAAAKg	FILE	Gamified v Traditional Feedback.xlsx		1762403136000	SYNCED
5	/photos/cyber-security-concept-data-…	/photos/cyber-security-concept-data-…	ENCODED_FILE_OBJ_ID	id:WCWiYh9PYuoAAAAAAAAAAEA	FILE	cyber-security-concept-data-security…		1762401079000	SYNCED
6	/work-FOLDER	/work	ENCODED_FILE_OBJ_ID	id:WCWiYh9PYuoAAAAAAAAAADw	FOLDER	Work		1761767848000	SYNCED

Additional insight comes from supporting databases such as prefs.db and pendingcomments.db. The DropboxAccountPrefs table in prefs.db contains key–value pairs describing account-level settings and state. Fields like ACCOUNT_INFO, FULL_ACCOUNT_INFO_V2 and PLAN_INFO_V2 hold opaque, likely encoded account structures, confirming that the emulator is linked to a valid Dropbox account rather than running in a logged-out state. Other preferences show user interface and notification behaviour: for example one DropboxAccountPrefs instance records that mention, comment, shared-content and task notifications are all enabled (NOTIFICATIONS_*_ENABLED = true), the last tab group used was BROWSER, and there is a record of the most recent “send-to” picker user ID. These settings suggest active, ongoing use of the app for collaboration rather than a one-off login.

DB Browser for SQLite - C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\dropbox_data\databases\

Table: DropboxAccountPrefs

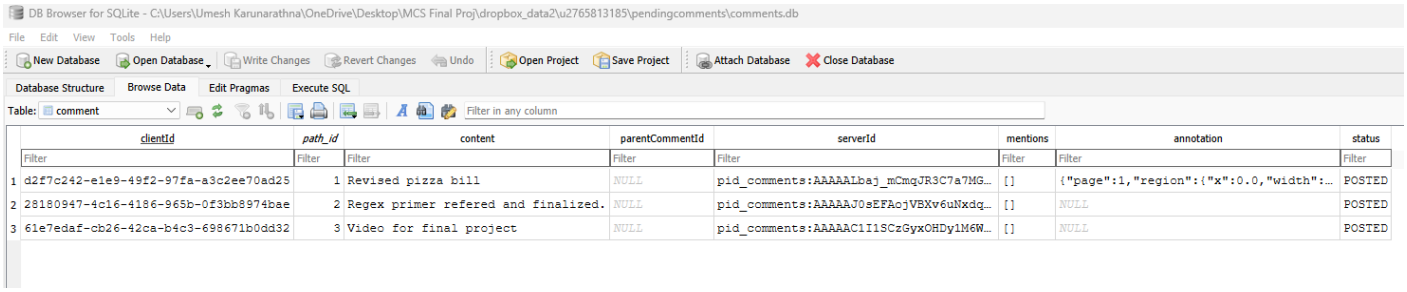
	_id	pref_name	pref_value
1	1	ACCOUNT_INFO	ClAKCjI3NjU4MTMxODUSFmNzaXM0NDQwdXNl…
2	2	USER_FEATURE_GATING_SDK_ENABLED	false
3	3	USER_FLAGS	EAEoADAA
4	4	SHOULD_SHOW_CU_ONBOARDING	false
5	5	OPEN_WITH_USER_DATA	CkYKBHhsc20QASIgChpjb20ubWljcm9zb2Z0…
6	6	DEVICE_LIMIT_DATA	CAAQAxgCIKHm7cOrMygA
7	7	CAMERA_UPLOAD_LOCKOUT_STATE	false
8	8	SHOULD_SHOW_LOCKOUT_ALERT	false
9	9	ALL_PHOTOS_CURSOR	Aij7OXHbbbMnuL1IiQawB8C7t7K51PcBT84v…
10	10	SEEN_LINK_DESKTOP_PROMPT	true
11	11	FULL_ACCOUNT_INFO_V2	CnIKKGRiaWQ6QUFBblgtMGU3Mzg3VERfMGpU…
12	12	PLAN_INFO_V2	CihkYmlkOkFBQW5YLTBlNzM4N1REXzBqVFBa…
13	13	LAST_URI	content://…
14	14	LIGHT_ALBUMS_DELTA_CURSOR	AASQrw-…
15	15	ALBUMS_DELTA_CURSOR	AATfPi3o7JvleggwFmmJcK8jp3JDdM-…

DB Browser for SQLite - C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final P

Table: DropboxAccountPrefs

	_id	pref_name	pref_value
1	1	NOTIFICATIONS_MENTION_ENABLED	true
2	2	NOTIFICATIONS_COMMENT_ENABLED	true
3	3	NOTIFICATIONS_SHARED_CONTENT_ENABLED	true
4	4	NOTIFICATIONS_TASK_ENABLED	true
5	5	SHARED_PROPERTIES_VERSION	1
6	6	LAST_REPORT_HOST_TIME	1764031664887
7	7	SHOULD_SHOW_PAIRING_INTRO	false
8	8	LAST_TAB_GROUP	BROWSER
9	9	LAST_TAB_GROUP_TIME	1764032461876
10	10	HAS_SHOWN_DRAWER_FIRST_TIME	true
11	11	CLIENT_LINK_REMINDER_BANNER_DISMISS_…	1
12	12	MOST_RECENT_PICKER_USER_SEND_TO	2765813185
13	13	MOST_RECENT_PICKER_TIME_SEND_TO	1764031984480
14	14	SEEN_DESTINATION_PICKER_COACHMARK	true

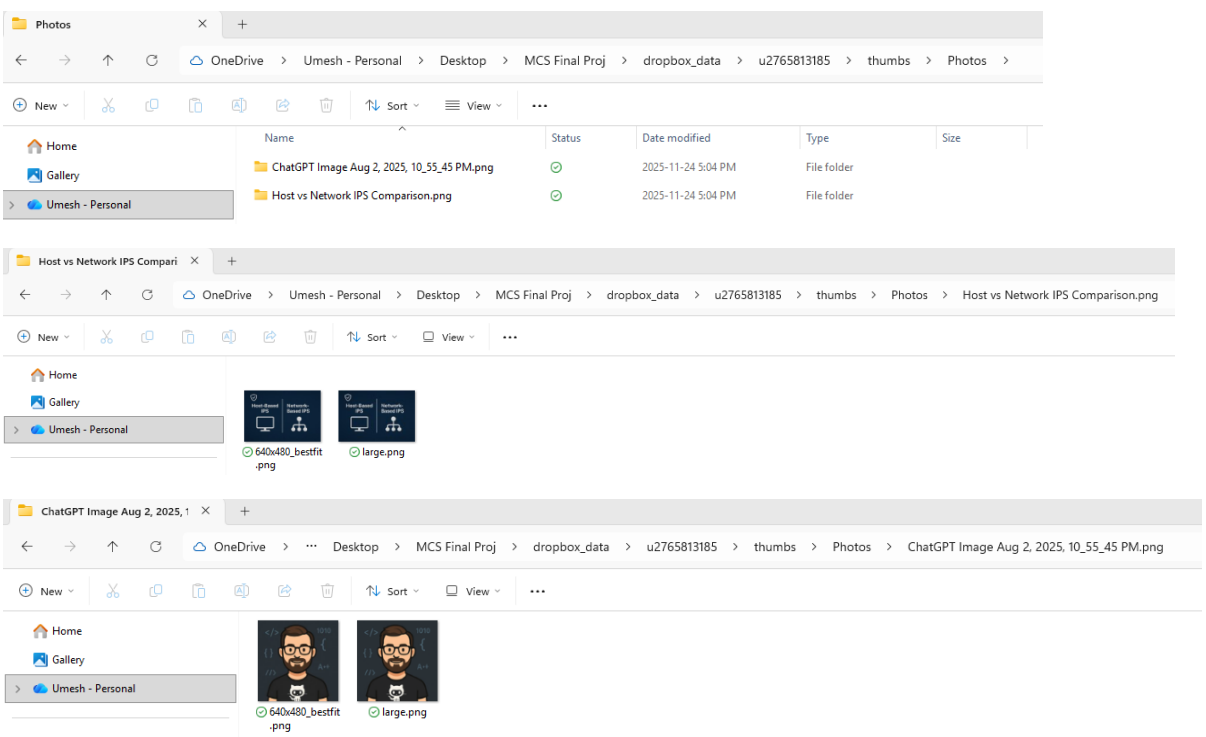
Finally, the comment table in pendingcomments.db shows three comments associated with PDF or image files, with content such as “Revised pizza bill”, “Regex primer referred and finalized.” and “Video for final project”. Even though the comment metadata is relatively minimal (client ID, path ID, status POSTED), it links specific shared documents to contextual notes, offering further evidence about how the user interacted with and revised their project material inside Dropbox.



	clientId	path_id	content	parentCommentId	serverId	mentions	annotation	status
1	d2f7c242-e1e9-49f2-97fa-a3c2ee70ad25	1	Revised pizza bill	NULL	pid_comments:AAAAALbaj_mCmqJR3C7a7MG…	[]	{"page":1,"region":{"x":0.0,"width":…	POSTED
2	28180947-4c16-4186-965b-0f3bb8974bae	2	Regex primer refered and finalized.	NULL	pid_comments:AAAAAJ0sEFAojVBXv6uNxdq…	[]	NULL	POSTED
3	61e7edaf-cb26-42ca-b4c3-698671b0dd32	3	Video for final project	NULL	pid_comments:AAAAAC1I1SCzGyxOHDy1M6W…	[]	NULL	POSTED

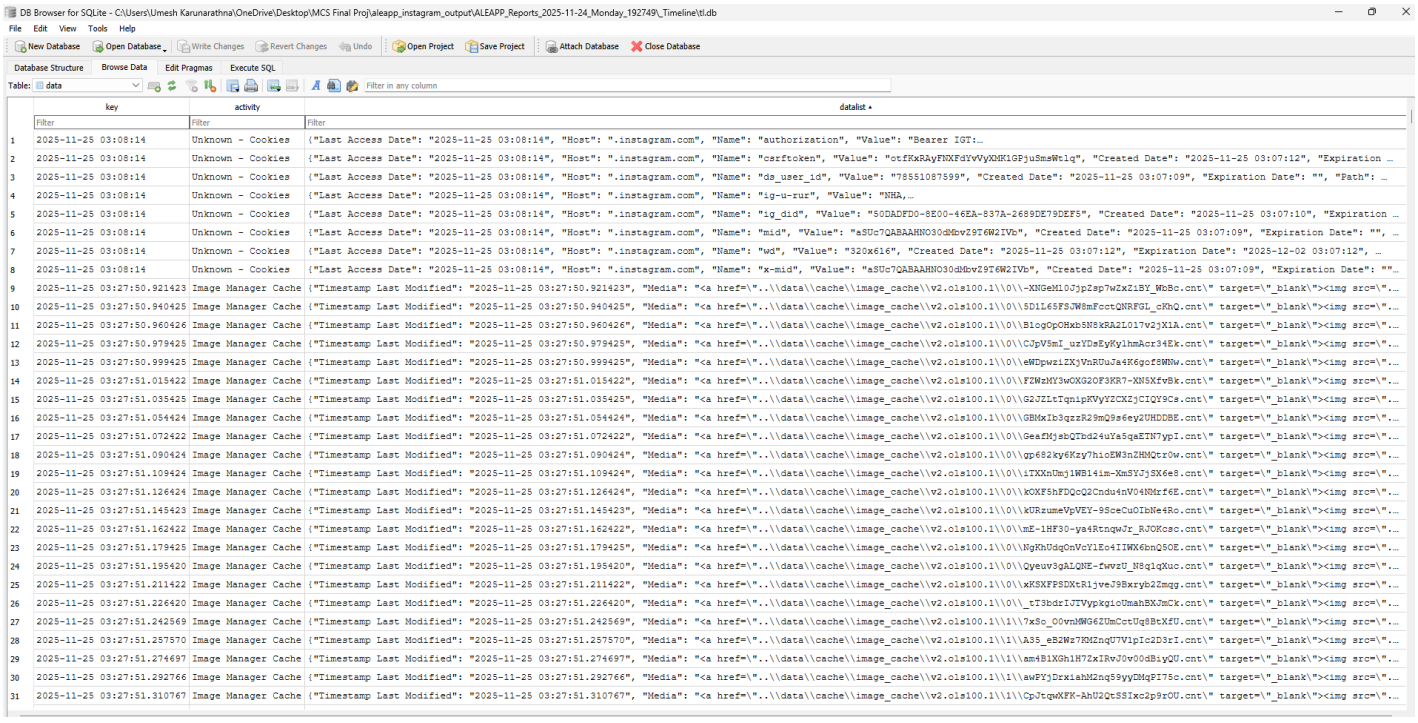
The extracted Dropbox data also contained a thumbs/Photos/ directory, which stores locally cached thumbnails for images saved in the user’s Dropbox account. Each photo has two versions, **large** and **640×480_bestfit** indicating Dropbox generates multiple resolutions for quick preview inside the mobile app. These thumbnails match the entries found earlier in the thumbnail_info table (e.g., same canonical paths such as /photos/host vs network ips… and /photos/chatgpt image aug), confirming that the app maintains local optimized copies for faster browsing.

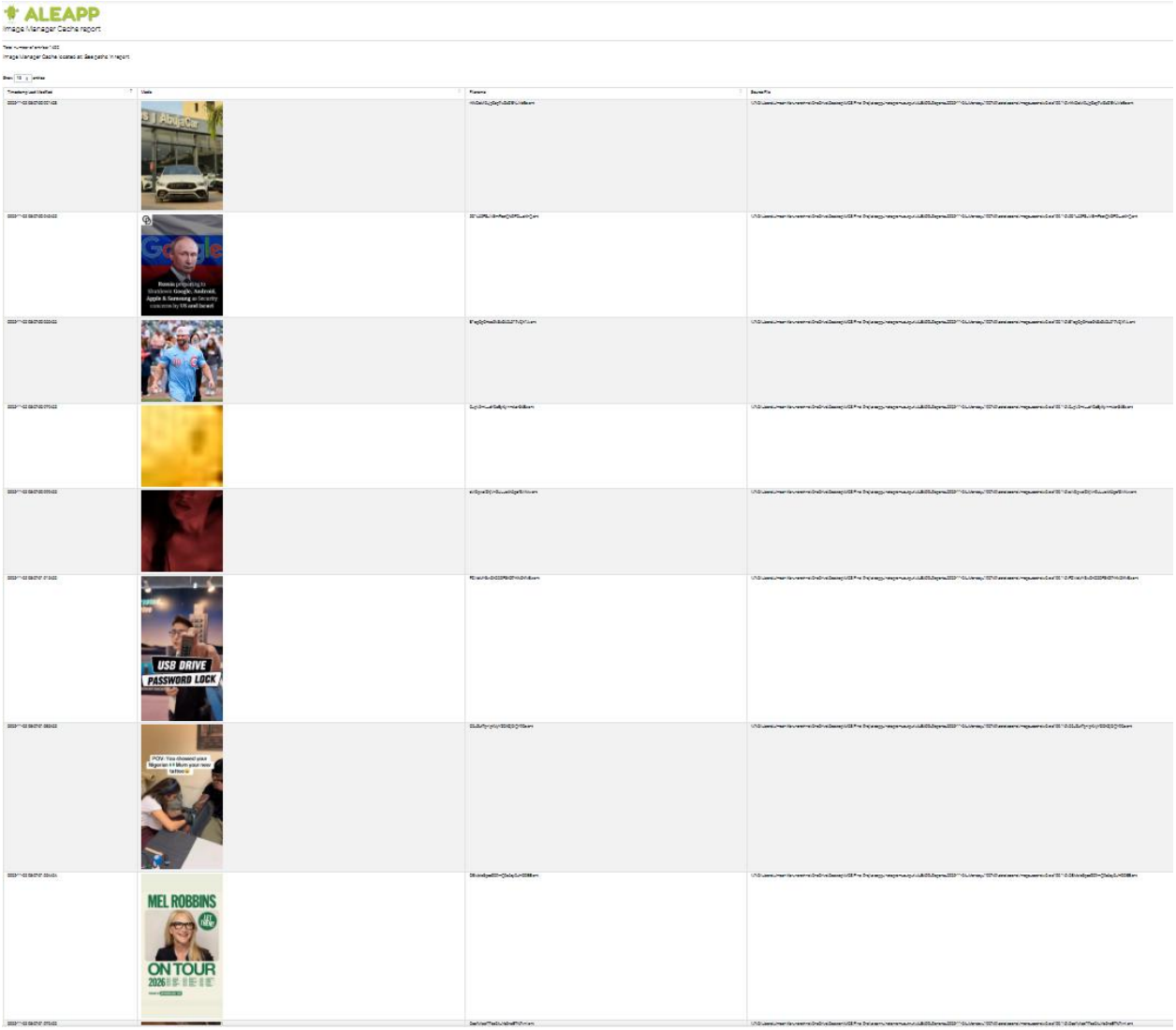
The recovered thumbnails further validate the presence of the corresponding media in the user’s Dropbox and represent an additional artifact category that examiners can use when full-resolution files are missing or deleted. The thumbnails also preserve the original filenames and folder structure, which helps reconstruct how the user organized files within Dropbox.



3.2 Instagram Lite

For Instagram Lite, the analysis began by extracting the entire */data/data/com.instagram.lite* directory from the emulator using ADB. The extracted files were then examined using DB Browser for SQLite, but only minimal structured data was available because Instagram Lite stores very little user-level information in its local databases. To perform a deeper forensic review, the dataset was processed using ALEAPP, which successfully parsed the application's internal cache directories. ALEAPP revealed a substantial **Image Manager Cache** containing 1,402 cached image files, representing thumbnails, profile photos, feed content, and other viewed media fragments these formed the primary recoverable artifacts from the app. ALEAPP's **Timeline view** also identified Instagram Lite activity, including cookies related to authorization, session identifiers, device tokens, and multiple entries referencing the cached image files. These timeline artifacts further confirm active usage of the app and show when specific cached items were accessed or generated during user interaction.





3.3 ALEAPP

3.3.1 ULR User Preferences

ALEAPP parsed the file ULR_USER_PREFS.xml from Google Play Services. This file stores the Google account's location-related settings. The report showed that the account *csis4440user@gmail.com* had all ULR options disabled, including location history, reporting, and auto-enable flags. A server sync timestamp (serverMillis) was also present, indicating the settings were recently updated.

ALEAPP

ULR User Preferences report

Total number of entries: 12

ULR User Preferences located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.gms\shared_prefs\ULR_USER_PREFS.xml

Show 15 entries

Search:

Name	Value
accountType_Account {name=csis4440user@gmail.com, type=com.google}	0
auto_enable_eligible_Account {name=csis4440user@gmail.com, type=com.google}	false
auto_enable_history_enabled_Account {name=csis4440user@gmail.com, type=com.google}	false
auto_enable_new_account_Account {name=csis4440user@gmail.com, type=com.google}	false
defined	true
historyEnabled_Account {name=csis4440user@gmail.com, type=com.google}	false
historyPackage_Account {name=csis4440user@gmail.com, type=com.google}	
reportingEnabled_Account {name=csis4440user@gmail.com, type=com.google}	false
reportingPackage_Account {name=csis4440user@gmail.com, type=com.google}	
restriction_Account {name=csis4440user@gmail.com, type=com.google}	0
serverMillis_Account {name=csis4440user@gmail.com, type=com.google}	1764031640253
updateNumber_Account {name=csis4440user@gmail.com, type=com.google}	2
Name	Value

3.3.2 Bluetooth Adapter Information

ALEAPP extracted Bluetooth configuration details from:

/data/misc/bluedroid/bt_config.conf

The artifact contained 10 key entries, providing low-level system metadata about the emulator's Bluetooth subsystem. The file includes the Bluetooth MAC address **(3c:5a:b4:01:02:03),** discovery timeout settings, and multiple cryptographic keys used for Bluetooth Low-Energy (BLE) operations, such as LE_LOCAL_KEY_DHK, LE_LOCAL_KEY_ER, LE_LOCAL_KEY_IR, and LE_LOCAL_KEY_IRK. These keys normally support secure pairing processes and device identity management. The ScanMode value was set to 0, indicating that the adapter was not configured for discoverable or pairing mode at the time of capture. The configuration also included a Salt256Bit value used in hashing operations. A TimeCreated timestamp (2025-11-25 00:39:49) shows when the configuration file was first generated. Overall, this artifact confirms that the emulator initialized a virtual Bluetooth adapter, but no paired devices, scan records, or connection history were present.

ALEAPP

Bluetooth Adapter Information report

Total number of entries: 10

Bluetooth Adapter Information located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\misc\bluedroid\bt_config.conf

Show 15 entries Search:

Key	Value
ScanMode	0
DiscoveryTimeout	120
TimeCreated	2025-11-25 00:39:49
LE_LOCAL_KEY_DHK	241a92abbacdc88a92c8f2f6439440ff
Address	3c:5a:b4:01:02:03
LE_LOCAL_KEY_IRK	77ec651b944ce88939795b30bf4a2b34
LE_LOCAL_KEY_IR	c6237b3267458b6b7348336669983c64
Salt256Bit	d3b4294a9ebe796252e4d1b8c20650cf3b8aa32e3060f3413d76af9f7a0015d7
FileSource	Empty
LE_LOCAL_KEY_ER	ff5c491951dcb074ec5855624a94e82a
Key	Value

3.3.3 Calendar (calendar.db)

ALEAPP extracted one calendar record from the built-in Android Calendar provider database located at:

/data/data/com.android.providers.calendar/databases/calendar.db

The artifact shows a single Google-linked calendar associated with the user account **csis4440user@gmail.com**. The calendar was created on 2025-11-25 00:49:07 UTC, and is marked as visible and primary, indicating that it is the default calendar for the user profile. The calendar's account type is listed as com.google, confirming synchronization with Google Calendar services. No additional metadata such as custom calendar names, event color categorization beyond the assigned color index (14), or geographic calendar location was present. This suggests the device contained only the default system-generated calendar with no additional user-created calendars.

ALEAPP

Calendar - Calendars report

Calendar - Calendars

Total number of entries: 1

Calendar - Calendars located at: \data\data\com.android.providers.calendar\databases\calendar.db

Show 15 entries Search:

Created Timestamp	Calendar Name	Calendar Display Name	Account Name	Account Type	Visible	Calendar Location	Timezone	Owner Account	Is Primary	Color	Color Index	Source
2025-11-25 00:49:07+00:00	csis4440user@gmail.com	csis4440user@gmail.com	csis4440user@gmail.com	com.google	Yes		UTC	csis4440user@gmail.com	Yes	-6299161	14	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.providers.calendar\databas
Created Timestamp	Calendar Name	Calendar Display Name	Account Name	Account Type	Visible	Calendar Location	Timezone	Owner Account	Is Primary	Color	Color Index	Source

3.3.4 Call Logs (calllog.db)

ALEAPP retrieved two call log entries from the device, sourced from the call log database located at:

/data/data/com.android.providers.contacts/databases/calllog.db

Both entries correspond to **outgoing calls** made to the same test number, **+1-555-521-5554**, which is typical for emulator environments. The calls occurred on **2025-11-25** at **05:59:22** and **06:01:47 UTC**, each lasting **5 seconds**. No incoming or missed calls were present, and no voicemail or transcription data was associated with the records. The country ISO field (US) reflects the emulator's default configuration. The absence of deleted entries or extended metadata indicates minimal user activity and aligns with a test-oriented device state.

ALEAPP

Call logs report

Total number of entries: 2

Call logs located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.providers.contacts\databases\calllog.db-journal

Show 15 entries

Search:

Call Date	Phone Account Address	Partner	Type	Duration in Secs	Partner Location	Country ISO	Data	Mime Type	Transcription	Deleted	Source File
2025-11-25 05:59:22	+15555215554	544258844	Outgoing	5		US				0	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.providers.contacts\databases\calllog.db
2025-11-25 06:01:47	+15555215554	655214785	Outgoing	5		US				0	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.providers.contacts\databases\calllog.db
Call Date	Phone Account Address	Partner	Type	Duration in Secs	Partner Location	Country ISO	Data	Mime Type	Transcription	Deleted	Source File

3.3.5 Google Chrome Data

ALEAPP extracted a substantial amount of browser-related artifacts from Google Chrome, making it one of the richest sources of user activity evidence on the device. The recovered datasets include bookmarks, cookies, downloads, keyword searches, offline pages, top sites, web history, and web visit logs. These artifacts collectively provide insight into user intent, browsing patterns, search behavior, and interaction timelines.

Chrome Bookmarks

The bookmarks database contained a single saved bookmark: Amazon.ca, added on 2025-11-25. This reflects intentional user action and indicates that the Amazon shopping site was of continued interest. Bookmark data is valuable in forensic investigations as it reveals curated preferences rather than accidental browsing.

ALEAPP

Chrome - Bookmarks report

Total number of entries: 1

Chrome - Bookmarks located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.chrome\app_chrome\Default\Bookmarks

Show 15 entries

Search:

Added Date	URL	Name	Parent	Type
2025-11-25 05:57:13.045965	https://www.amazon.ca/	Amazon.ca: Low Prices â€" Fast Shipping â€" Millions of Items	Mobile bookmarks	url
Added Date	URL	Name	Parent	Type

Chrome Cookies

Chrome's cookie store was one of the largest artifacts with 342 entries. These cookies included session tokens, tracking identifiers, and long-term authentication artifacts for Google, YouTube, and other major services. Many cookies had expiration dates extending

to 2027, showing they were persistent rather than session-only. Such cookies can reveal login activity, account usage timelines, and potential cross-service authentication presence. The volume of cookies indicates repeated browsing sessions and interaction with Google services through both Chrome and embedded WebView components.

ALEAPP

Chrome - Cookies report

Total number of entries: 342

Chrome - Cookies located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.chrome\app_chrome\Default\Cookies

Show 15 entries

Search:

Last Access Date	Host	Name	Value
2025-11-25 04:30:07	.pippio.com	nnls	
2025-11-25 05:55:58	.bestbuy.ca	dtSacfbhlec8	-
2025-11-25 04:30:06	.mfadsrvr.com	ssh	!medianet=1764045007
2025-11-25 04:30:43	.analytics.yahoo.com	IDSYNC	"19gg~2ss4:19gl~2ss4:19h2~2ss4:19h3~2ss4"
2025-11-25 05:56:21	.crwdcntrl.net	_cc_aud	"ABR4nGNgYGDIVHVY4ABJgbmk79AD0aTP4EkADNtBGc%3D"
2025-11-25 05:56:21	.crwdcntrl.net	_cc_cc	"ACZ4nGNQsLBMtkxOSU4zNDZONTJJMjMzSTJOTU00N7FINTE1M09mAIJMVVc2BjjgXT%2FzrgjjXX%2BG%2F4yMDOdOPWKDsS8vnsMCY79c%2B5Qbxp58Qh2u%2FOghZhj73WdLGPPgsqIwnZuRTNn0pxDG%2FLtxClx42rf%2FKjD2yj8bdWB
2025-11-25 04:31:56	.linkedin.com	lidc	"b=VGST02:s=V:r=V:a=V:p=V:g=3672:u=1:x=1:i=1764045009:t=1764131409:v=2:sig=AQELV77OWZWmUBH0LuDDVrfzB3vPtSmuJ"
2025-11-25 04:31:56	.linkedin.com	bcookie	"v=2&5073ee74-f853-4295-8a6c-29a8640a1d66"
2025-11-25 04:45:21	.uptodown.com	_pk_ref.2.866e	%5B%22%22%2C%22%22%2C1764045002%2C%22https%3A%2F%2Fwww.google.com%2F%22%5D
2025-11-	.uptodown.com	FCNEC	%5B%5B%22AKsRol_yyhWMEpcgl4MuFVQh9G-2wkCUltTZ22Sls_UUdCADk5swRwBcZ3KtaH7mRiC1gaff3nqsIEPWRQKzwuLRWidf7wNGXmOL0UWrAki6KIpw9ySp-yICvb6ggKuGrP43OzFBvr3geY2efDBb6SknOTGFJYi9KQ%3D%3D%22%5D%

Chrome Downloads

The downloads table contained one aborted download from Uptodown, linked to obtaining an older Instagram APK. The attempted download was ~88MB and marked as “Canceled”. This matches the user action of installing older Instagram versions for compatibility. Download history is a strong indicator of software acquisition attempts and can be used to correlate app installations with web behaviors.

ALEAPP

Chrome - Downloads report

Total number of entries: 1

Chrome - Downloads located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.chrome\app_chrome\Default\History

Show 15 entries

Search:

Start Time	End Time	Last Access Time	URL	Target Path	State	Danger Type	Interrupt Reason	Opened?	Received Bytes	Total Bytes
2025-11-25 04:45:22			https://dw.uptodown.com/dwn/isBCARastyZwxsEzLOjdzA6Sor9Sr2cJpLYnxFljd8UmaHExx3c4gk9DJNuF7rGL16GGoAmPDRPBNA33QWZ-4Q8Ftm5Ob0Jh5l7R2-JtOyeq_zj3cw40Rh6rzwc_13tl/n9y9VtR79yFGqSf5TI1W-AkF1injShFwBggw_w0ongp1bk03JfMn3Y9eTL5pr9tMnqGHBY3MMNlpHbxu3y0wP3i3xfPo1VVCuPMngW_0zbtFN-QeBbBgJhfzBxivhg4w/SunidHGJGOi9is-Yy18htVpAkVxPdmM7utL_gW50fGf-B9dxPLKOIPmxg4nj7GleBdqP00PvMzAJKm6ajiQcYg==/		Canceled		Canceled		0	88989197
Start Time	End Time	Last Access Time	URL	Target Path	State	Danger Type	Interrupt Reason	Opened?	Received Bytes	Total Bytes

Chrome Keyword Search Terms

ALEAPP recovered 10 keyword search entries, clearly reflecting user search behavior. The earliest searches focused heavily on “cybersecurity”, suggesting research activity or topic exploration. Later queries involved locating older Instagram versions, AppStore++, and Play Store, matching the attempts to sideload apps on the emulator. These entries demonstrate both research intent and app installation troubleshooting patterns.

ALEAPP

Chrome - Keyword Search Terms report

Total number of entries: 10

Chrome - Keyword Search Terms located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.chrome\app_chrome\Default\History

Show 15 entries Search:

Last Visit Time	Term	URL
2025-11-25 00:50:15	cybersecurity	https://www.google.com/search?q=cybersecurity&oq=cybersecurity+&aqs=chrome..69i57j0l4.4118j0j7&client=ms-unknown&sourceid=chrome-mobile&ie=UTF-8
2025-11-25 00:50:16	cybersecurity	https://www.google.com/search?q=cybersecurity&oq=cybersecurity+&aqs=chrome..69i57j0l4.4118j0j7&client=ms-unknown&sourceid=chrome-mobile&ie=UTF-8&sei=R_0kadOHOP2q0PEP4-DqWQ
2025-11-25 00:51:03	cybersecurity	https://www.google.com/search?q=cybersecurity&client=ms-unknown&sca_esv=291655ba3e8774d2&tbm=isch&sxsrf=AE3TifPkfwEqFqd0Izy9:veuTtzKrWDZVg:1764031817470&source=lnms&sa=X&ved=0ahUKEwim7-bOioyRAxVgFjQIHf6xCg0Q_AUIBygC&biw=320&bih=560#imgrc=RZBSmrNdT_fqoM
2025-11-25 00:51:04	cybersecurity	https://www.google.com/search?q=cybersecurity&client=ms-unknown&sca_esv=291655ba3e8774d2&tbm=isch&sxsrf=AE3TifPkfwEqFqd0Izy9:veuTtzKrWDZVg:1764031817470&source=lnms&sa=X&ved=0ahUKEwim7-bOioyRAxVgFjQIHf6xCg0Q_AUIBygC&biw=320&bih=560
2025-11-25 00:51:31	cybersecurity	https://www.google.com/search?q=cybersecurity&client=ms-unknown&sca_esv=291655ba3e8774d2&tbm=isch&sxsrf=AE3TifPkfwEqFqd0Izy9:veuTtzKrWDZVg:1764031817470&source=lnms&sa=X&ved=0ahUKEwim7-bOioyRAxVgFjQIHf6xCg0Q_AUIBygC&biw=320&bih=560#imgrc=udyQSKivRYrFsM
2025-11-25 04:30:44	Instagram older version uptodown	https://www.google.com/search?q=Instagram+older+version+uptodown&oq=Instagram+older+version+uptodown &aqs=chrome..69i57j0l4.29831j0j1&client=ms-unknown&sourceid=chrome-mobile&ie=UTF-8
2025-11-25 04:45:33	appstore++	https://www.google.com/search?q=appstore%2B%2B&oq=appstore%2B%2B&aqs=chrome..69i57j0l4.3236j0j4&client=ms-unknown&sourceid=chrome-mobile&ie=UTF-8
2025-11-25 04:45:40	appstore	https://www.google.com/search?client=ms-unknown&sca_esv=9a662cfbafe167d5&sxsrf=AE3TifN8YsHbDPfrJySiJ:Ve1yQ2XGqSJXA%3A1764045935009&q=appstore&oq=appstore%2B%2B&aqs=heirloom-srp..
2025-11-25 04:45:50	playstore	https://www.google.com/search?q=playstore&oq=playstore&aqs=chrome..69i57j0l4.2065j0j4&client=ms-unknown&sourceid=chrome-mobile&ie=UTF-8
2025-11-25 05:55:55	Dji	https://www.bestbuy.ca/en-ca/search?search=Dji

Chrome Offline Pages

Chrome had saved one offline page an offline copy of Amazon.ca indicating the page was either manually saved or automatically cached for offline access. Offline files represent preserved snapshots of web content and can contain artifacts not visible through browsing history alone.

ALEAPP

Chrome - Offline Pages report

Total number of entries: 1

Chrome - Offline Pages located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.chrome\app_chrome\Default\Offline Pages\metadata\OfflinePages.db

Show 15 entries Search:

Creation Time	Last Access Time	Online URL	File Path	Title	Access Count	File Size
2025-11-25 05:57:13	2025-11-25 05:57:13	https://www.amazon.ca/	/data/user/0/com.android.chrome/cache/Offline Pages/archives/aea61cb6-7301-4876-92c2-5c6b88521b09.mhtml	Amazon.ca: Low Prices – Fast Shipping – Millions of Items	0	1408007
Creation Time	Last Access Time	Online URL	File Path	Title	Access Count	File Size

Chrome Top Sites

The Top Sites database identified the most frequently accessed domains:

- BestBuy Canada
- Instagram
- Amazon
- CBC

These match the browsing history and reinforce the pattern of shopping, social media access, and general browsing.

ALEAPP

Chrome - Top Sites report

Total number of entries: 4

Chrome - Top Sites located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.chrome\app_chrome\Default\Top Sites

Show 15 entries

URL	Rank	Title
http://www.cbc.ca/	1	CBC.ca - watch, listen, and discover with Canada's Public Broadcaster
https://m.bestbuy.ca/	0	Best Buy: Shop Online For Deals & Save | Best Buy Canada
https://www.amazon.ca/	3	Amazon.ca: Low Prices – Fast Shipping – Millions of Items
https://www.instagram.com/	2	Instagram
URL	Rank	Title

Chrome Web History and Visit Logs

The browser history included 38 web history entries and 64 web visit records, showing detailed timestamps, navigation patterns, and transition types (e.g., link, redirect, forward/back). Activity included:

- Extensive viewing of cybersecurity pages and Google Image Search results
- Searches related to APK downloads
- Visits to BestBuy, Instagram, Amazon, and CBC
- Multiple redirect chains from Google Search, indicating interactive browsing

These artifacts are particularly valuable because they reconstruct the timeline of user actions with minute-by-minute accuracy. ALEAPP's timeline also merges these events with other device artifacts, allowing correlation with app installations, call logs, and system events.

ALEAPP

Chrome - Web History report

Total number of entries: 38

Chrome - Web History located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.chrome\app_chrome\Default\History

Show 15 entries Search:

Last Visit Time	URL	Title	Visit Count	Typed Count	ID	Hidden
2025-11-25 00:50:15	https://www.google.com/search?q=cybersecurity&oq=cybersecurity+&aqs=chrome..69i57j0l4.4118j0j7&clien t=ms-unknown&sourceid=chrome-mobile&ie=UTF-8	cybersecurity - Google Search	1	0	1	
2025-11-25 00:50:16	https://www.google.com/search?q=cybersecurity&oq=cybersecurity+&aqs=chrome..69i57j0l4.4118j0j7&clien t=ms-unknown&sourceid=chrome-mobile&ie=UTF-8&sei=R_0kadOHOP2q0PEP4-DqWQ	cybersecurity - Google Search	2	0	2	
2025-11-25 00:50:55	https://utimaco.com/news/blog-posts/importance-cybersecurity	What is cybersecurity and why is it important? - Utimaco	1	0	5	
2025-11-25 00:51:03	https://www.google.com/search?q=cybersecurity&client=ms-unknown&sca_esv=291655ba3e8774d2&tbm=isch&sx srf=AE3TifPkfwEqFqd0Izy9-veuTtzKrWDZVg:1764031817470&source=lnms&sa=X&ved=0ahUKEwim7-bOioyRAxVgFjQIHf6xCg0Q_AUIBygC&biw=320&bih=560#imgrc=RZBSmrNdT_fqoM	cybersecurity - Google Search	4	0	4	
2025-11-25 00:51:04	https://www.google.com/search?q=cybersecurity&client=ms-unknown&sca_esv=291655ba3e8774d2&tbm=isch&sx srf=AE3TifPkfwEqFqd0Izy9-veuTtzKrWDZVg:1764031817470&source=lnms&sa=X&ved=0ahUKEwim7-bOioyRAxVgFjQIHf6xCg0Q_AUIBygC&biw=320&bih=560	cybersecurity - Google Search	4	0	3	
2025-11-25 00:51:16	https://builtin.com/cybersecurity	What Is Cybersecurity? | Built In	1	0	7	
2025-11-25 00:51:31	https://www.google.com/search?q=cybersecurity&client=ms-unknown&sca_esv=291655ba3e8774d2&tbm=isch&sx srf=AE3TifPkfwEqFqd0Izy9-veuTtzKrWDZVg:1764031817470&source=lnms&sa=X&ved=0ahUKEwim7-bOioyRAxVgFjQIHf6xCg0Q_AUIBygC&biw=320&bih=560#imgrc=udyQSKivRYrFsM	cybersecurity - Google Search	2	0	6	
2025-11-25	https://www.google.com/search?q=Instagram+older+version+uptodown&oq=Instagram+older+version+uptodown &aqs=chrome..69i57j0l4.29831j0j1&client=ms-unknown&sourceid=chrome-mobile&ie=UTF-8	Instagram older version uptodown - Google Search	4	0	9	

Unknown WebView Cookies

ALEAPP also detected 23 cookies associated with Google QuickSearchBox WebView. These included authentication cookies for accounts.google.com and Google services. This reveals that the Google Search app and embedded WebView components also performed authenticated browsing or login activity outside Chrome itself. Such cross-application cookies help identify the presence of account syncing or shared authentication.

ALEAPP

Unknown - Cookies report

Total number of entries: 23

Unknown - Cookies located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.googlequicksearchbox\app_webview\Default\Cookies

Show 15 entries

Search:

Last Access Date	Host	Name	Value	Created Date	Expiration Date
2025-11-25 00:47:11	accounts.google.com	ACCOUNT_CHOOSER	AFx_qI6c86mp1WSn7Mc8un0DwfQX0tQf15eK2hZJQd9X5LaAiyGDsSBzbUsccN3VG20bchmSd6xk1p8Dsd_tAwren9ypVtw8G5fvarBs7kGgVldti1z2riA6s-DV-Opk1THzfK05zjofRHugkJo039RdhYYN1iav8w	2025-11-25 00:47:11	2027-11-25 00:47:11
2025-11-25 00:47:11	.google.ca	APISID	Rv93aJ4Sp97WVoae/AxOBXeo5RMZGuzECN	2025-11-25 00:47:11	2027-11-25 00:47:11
2025-11-25 00:47:11	.google.com	APISID	Rv93aJ4Sp97WVoae/AxOBXeo5RMZGuzECN	2025-11-25 00:47:11	2027-11-25 00:47:11
2025-11-25 00:47:11	.google.ca	HSID	A59thSp4xpKMohC9B	2025-11-25 00:47:11	2027-11-25 00:47:11
2025-11-25 00:47:11	.google.com	HSID	AdV6RvSHKtTJgDP_P	2025-11-25 00:47:11	2027-11-25 00:47:11
2025-11-25 00:47:11	accounts.google.com	LSID	s.CA:g.a0003wg9P3M1AhujOQj4cy1qX0LwceRtKPMzzok01kcbgXT00xUu3bw0UPHIJ1jUy8X0MMJHdwACgYKAaQSARESFQHGX2MistIWI2gxixLUTcY558ARAhoVAUF8yKpP_n7ZMsuQZpC3WXVVXcJZ0076	2025-11-25 00:47:11	2027-11-25 00:47:11
2025-11-25 00:47:11	.google.ca	SAPISID	moXgSuQ8kCuBsEWS/ALIMIwRvfVXimcsJr	2025-11-25 00:47:11	2027-11-25 00:47:11
2025-11-25 00:47:11	.google.com	SAPISID	moXgSuQ8kCuBsEWS/ALIMIwRvfVXimcsJr	2025-11-25 00:47:11	2027-11-25 00:47:11

3.3.4 Contacts

ALEAPP extracted three contact entries from the device's contacts2.db database, indicating that the emulator had a minimal but functional contacts list configured. Each contact entry contained a display name and a phone number, but no associated email addresses, profile photos, or additional metadata fields. The contacts identified were **Shakira**, **Shane**, and **User1**, each with a unique phone number formatted in a North American style. These entries help demonstrate how Android stores and manages contact data at the database level using the vnd.android.cursor.item/phone_v2 MIME type.

Although these entries appear to be test or dummy contacts, their presence confirms the integrity of the contacts provider and validates that ALEAPP was able to correctly parse structured fields such as names, numbers, and MIME associations. This artifact also complements related evidence such as call logs; for instance, call history entries referencing numbers present in the contacts database can establish linkage between call events and specific identities. While the dataset is small, it demonstrates the standard storage structure of contacts and showcases ALEAPP's ability to extract and contextualize personal identity information from Android system databases.

ALEAPP

Contacts report

Contacts from the device

Total number of entries: 3

Contacts located at: See source file(s) below

Show 15 entries

Mimetype	Data 1	Display Name	Phone Number	Email Address	Source File
vnd.android.cursor.item/phone_v2	(655) 214-785	Shakira	(655) 214-785		\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.providers.contacts\databases\contacts2.db
vnd.android.cursor.item/phone_v2	(455) 566-774	Shane	(455) 566-774		\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.providers.contacts\databases\contacts2.db
vnd.android.cursor.item/phone_v2	(345) 333-222	User1	(345) 333-222		\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.providers.contacts\databases\contacts2.db
Mimetype	Data 1	Display Name	Phone Number	Email Address	Source File

3.3.4 Sim Information

The Device Info artifact extracted from telephony.db provides essential identifiers and network-related metadata associated with the emulator's SIM and carrier configuration. ALEAPP parsed a single entry that includes the device's assigned phone number (**+15555215554**), the IMSI (**310260000000000**), the ICC ID, carrier name (**T-Mobile**), and the linked ISO country code (**us**). These values are characteristic of Android emulator defaults, confirming that the device uses a simulated T-Mobile network profile typically embedded into AOSP-based virtual environments.

Despite originating from an emulator rather than a physical SIM card, this information remains forensically valuable because it demonstrates how Android stores telephony identifiers in com.android.providers.telephony databases. The persistence of the phone number and IMSI in plaintext allows investigators to map communication events (e.g., SMS, call logs) back to specific subscriber identities. The ICC ID additionally reveals a structured representation of the virtual SIM metadata, showing consistent formatting with real-world SIM cards. This artifact therefore helps validate the linkage between telephony services and other communication logs found on the device, illustrating ALEAPP's ability to extract deeper network-level metadata beyond app-specific evidence.

ALEAPP

Device Info report

Total number of entries: 1

Device Info located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\user_de\0\com.android.providers.telephony\databases\telephony.db

Show 15 entries

Search:

Number	IMSI	Display Name	Carrier Name	ISO Code	Carrier ID	ICC ID
+15555215554	310260000000000	T-Mobile	T-Mobile	us	1	89014103211118510720
Number	IMSI	Display Name	Carrier Name	ISO Code	Carrier ID	ICC ID

3.3.5 Settings - Secure

The settings_secure.xml artifact revealed four important system identifiers: the Android ID, Bluetooth MAC address, Bluetooth device name, and the mock location flag. The **Android ID** (4d528c81f15ed725) uniquely identifies this installation of the device and helps correlate data across apps. The **Bluetooth address** (3C:5A:B4:01:02:03) and device name (sdk_gphone_x86_64) confirm that the evidence originated from an Android emulator rather than a physical device. Finally, **mock_location = 0** indicates that no fake GPS or location spoofing was enabled, meaning any location artifacts in the broader dataset can be considered forensically reliable.

ALEAPP

Settings Secure report

Total number of entries: 4

Settings Secure located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\system\users\0\settings_secure.xml

Show 15 entries

Name	Value
android_id	4d528c81f15ed725
bluetooth_address	3C:5A:B4:01:02:03
bluetooth_name	sdk_gphone_x86_64
mock_location	0
Name	Value

3.3.5 Digital Wellbeing (App Usage Events)

The Digital Wellbeing event log contained **634 timestamped activity entries**, offering a detailed reconstruction of how the device was used over time. These events showed system-level actions such as *DEVICE_SHUTDOWN* and *DEVICE_STARTUP*, along with app-level transitions like *ACTIVITY_RESUMED*, *PAUSED*, and *STOPPED* for various packages. This data is valuable because it provides a chronological timeline of user behaviour starting immediately after the emulator boot. It confirms which apps were opened (e.g., Settings, Setup Wizard), when notifications were fired, and how often apps became active or inactive. These patterns allow investigators to validate user interactions, correlate app openings with other evidence, and build a high-accuracy device usage timeline.

ALEAPP

Digital Wellbeing - Events report

Total number of entries: 634

Digital Wellbeing - Events located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.wellbeing\databases\app_usage

Show 15 entries Search:

Timestamp	Package ID	Event Type	Source File
2025-11-25 00:40:15+00:00	android	DEVICE_SHUTDOWN	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.wellbeing\databases\app_usage
2025-11-25 00:40:15+00:00	com.android.settings	ACTIVITY_RESUMED	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.wellbeing\databases\app_usage
2025-11-25 00:40:15+00:00	com.android.settings	ACTIVITY_PAUSED	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.wellbeing\databases\app_usage
2025-11-25 00:40:15+00:00	com.google.android.setupwizard	ACTIVITY_RESUMED	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.wellbeing\databases\app_usage
2025-11-25 00:40:15+00:00	com.google.android.setupwizard	ACTIVITY_PAUSED	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.wellbeing\databases\app_usage
2025-11-25 00:40:15+00:00	com.android.settings	ACTIVITY_RESUMED	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.wellbeing\databases\app_usage
2025-11-25 00:40:15+00:00	com.android.settings	ACTIVITY_STOPPED	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.wellbeing\databases\app_usage
2025-11-25 00:40:15+00:00	com.google.android.setupwizard	ACTIVITY_STOPPED	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.wellbeing\databases\app_usage
2025-11-25 00:40:15+00:00	android	NOTIFICATION	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.wellbeing\databases\app_usage
2025-11-25	android	DEVICE_STARTUP	\\?\C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-

3.3.6 Emulated Storage

The emulated storage artifacts showed clear evidence of user-generated and downloaded files. The **Downloads** entries captured images saved through Chrome and the APK file for Instagram Lite, including timestamps, URLs, and file sizes indicating active browsing and download activity. The **Files** report revealed camera-captured photos, Instagram-exported media, and a large TAR archive, showing varied user interactions with device storage. The **Images** section confirmed all stored pictures with metadata such as orientation, owner app, and file paths, helping validate when the files were created or acquired.

ALEAPP

Emulated Storage Metadata - Downloads report

Total number of entries: 3

Emulated Storage Metadata - Downloads located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.providers.media.module\databases\external.db-wal

Show 15 entries Search:

Key Timestamp	Date Added	Date Modified	Date Taken	Path	Title	Display Name	Size	Owner Package Name	Bucket Display Name	Referer URI	Downloa
2025-11-25 00:51:28+00:00	2025-11-25 00:51:28+00:00	2025-11-25 00:51:28+00:00		/storage/emulated/0/Download/cybersecurity.png	cybersecurity	cybersecurity.png	47677	com.android.chrome	Download	https://builtin.com/cybersecurity	https://c
2025-11-25 00:52:05+00:00	2025-11-25 00:52:05+00:00	2025-11-25 00:52:05+00:00		/storage/emulated/0/Download/13_P-600x337.png	13_P-600x337	13_P-600x337.png	328412	com.android.chrome	Download	https://kvmce.in/cyber-security/	https://k
2025-11-25 04:32:28+00:00	2025-11-25 04:32:28+00:00	2025-11-25 04:32:51+00:00		/storage/emulated/0/Download/instagram-379-1-0-43-80.apk	instagram-379-1-0-43-80	instagram-379-1-0-43-80.apk	88989197	com.android.chrome	Download	https://instagram.en.uptodown.com/	https://d 4Q8Frtm AkF1injS QeBbBg
Key Timestamp	Date Added	Date Modified	Date Taken	Path	Title	Display Name	Size	Owner Package Name	Bucket Display Name	Referer URI	Downloa

ALEAPP

Emulated Storage Metadata - Files report

Total number of entries: 24

Emulated Storage Metadata - Files located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.providers.media.module\databases\external.db-wal

Show 15 entries

Search:

Key Timestamp	Date Added	Date Modified	Date Taken	Path	Title	Display Name	Size	Latitude	Longitude	Orientation	Owner Package Name	Bucket Display Name	Referer URI	Download URI	Relative Path	Is Downloaded?
2025-11-25 00:38:15+00:00		2025-11-25 00:38:15+00:00		/storage		storage				Vertical						
2025-11-25 00:40:19+00:00		2025-11-25 00:40:19+00:00		/storage/emulated		emulated				Vertical						
2025-11-25 00:40:20+00:00		2025-11-25 00:40:20+00:00		/storage/emulated/0		0				Vertical					/	
2025-11-25 00:40:20+00:00		2025-11-25 00:40:20+00:00		/storage/emulated/0/Music		Music				Vertical					/	
2025-11-25 00:40:20+00:00		2025-11-25 00:40:20+00:00		/storage/emulated/0/Podcasts		Podcasts				Vertical					/	
2025-11-25 00:40:20+00:00		2025-11-25 00:40:20+00:00		/storage/emulated/0/Ringtones		Ringtones				Vertical					/	
2025-11-25 00:40:20+00:00		2025-11-25 00:40:20+00:00		/storage/emulated/0/Alarms		Alarms				Vertical					/	
2025-11-25 00:40:20+00:00		2025-11-25 00:40:20+00:00		/storage/emulated/0/Notifications		Notifications				Vertical					/	
2025-11-25 00:40:20+00:00		2025-11-25 00:40:20+00:00		/storage/emulated/0/Pictures		Pictures				Vertical					/	
2025-11-25		2025-11-25		/storage/emulated/0/Movies		Movies				Vertical					/	

ALEAPP

Emulated Storage Metadata - Images report

Total number of entries: 5

Emulated Storage Metadata - Images located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.providers.media.module\databases\external.db-wal

Show 15 entries

Search:

Key Timestamp	Date Added	Date Modified	Date Taken	Path	Title	Display Name	Size	Latitude	Longitude	Orientation	Owner Package Name	Bucket Display Name
2025-11-25 00:49:47+00:00	2025-11-25 00:49:47+00:00	2025-11-25 00:49:47+00:00		/storage/emulated/0/Pictures/IMG_20251124_164946.jpg	IMG_20251124_164946	IMG_20251124_164946.jpg	37506			Horizontal	com.android.camera2	Pictures
2025-11-25 00:49:49+00:00	2025-11-25 00:49:49+00:00	2025-11-25 00:49:49+00:00		/storage/emulated/0/Pictures/IMG_20251124_164949.jpg	IMG_20251124_164949	IMG_20251124_164949.jpg	38585			Horizontal	com.android.camera2	Pictures
2025-11-25 00:51:28+00:00	2025-11-25 00:51:28+00:00	2025-11-25 00:51:28+00:00		/storage/emulated/0/Download/cybersecurity.png	cybersecurity	cybersecurity.png	47677			Vertical	com.android.chrome	Download
2025-11-25 00:52:05+00:00	2025-11-25 00:52:05+00:00	2025-11-25 00:52:05+00:00		/storage/emulated/0/Download/13_P-600x337.png	13_P-600x337	13_P-600x337.png	328412			Vertical	com.android.chrome	Download
2025-11-25 03:16:05+00:00	2025-11-25 03:16:05+00:00	2025-11-25 03:16:05+00:00		/storage/emulated/0/Pictures/Instagram/1764040565437.jpg	1764040565437	1764040565437.jpg	54597			Vertical		Instagram
Key Timestamp	Date Added	Date Modified	Date Taken	Path	Title	Display Name	Size	Latitude	Longitude	Orientation	Owner Package Name	Bucket Display Name

3.3.7 Google Maps – Labeled Places

The Google Maps labeled places database revealed two user-defined locations: **Home** and **Work**, each containing accurate latitude/longitude coordinates, full postal addresses, and auto-generated Google Maps URLs. These entries indicate that the user manually saved frequently visited locations, which can be used to infer habitual movement patterns and points of interest. The precise timestamps show when each label was created, supporting timeline reconstruction and behavioral profiling.

ALEAPP

Google Maps Label Places report

Total number of entries: 2

Google Maps Label Places located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.maps\databases\gmm_myplaces.db

Show 15 entries Search:

Timestamp	Label	Latitude	Longitude	Address	URL
2025-11-25 05:58:11.022000+00:00	Home	49.227478	-123.0367	3278 E 47th Ave, Vancouver, BC V5S 1C6	http://maps.google.com/?q=3278+E+47th+Ave,+Vancouver,+BC+V5S+1C6&ftid=0x54867685aee7967d:0xcf5d6747c30f37cf
2025-11-25 05:58:41.817000+00:00	Work	49.225902	-122.998551	Decathlon, 4800 Kingsway unit 288, Burnaby, BC V5H 4P1	http://maps.google.com/?q=Decathlon,+4800+Kingsway+unit+288,+Burnaby,+BC+V5H+4P1&ftid=0x548677d463350725:0xfcc58165a209bafa
Timestamp	Label	Latitude	Longitude	Address	URL

3.3.8 Gmail Active Account

The Gmail shared preferences file shows that the device is actively logged into a single Google account: **csis4440user@gmail.com**. This confirms the primary identity associated with email services and other linked Google applications on the device. Even though no email content is extracted, this artifact is important because it establishes the **active user profile**, supports account attribution, and helps correlate findings across Chrome, Google Maps, Play Store activity, and other Google-linked artifacts.

ALEAPP

Gmail - Active report

Total number of entries: 1

Gmail - Active located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.gm\shared_prefs\Gmail.xml

Show 15 entries

Active Gmail Address	
csis4440user@gmail.com	
Active Gmail Address	

3.3.9 Google Photos Cache

The Google Photos cache database contains several cached image entries generated as thumbnails or temporary files during photo viewing and synchronization. Although the filenames are hashed, the records include timestamps and file sizes which indicate recent image activity around **2025-11-25**, aligning with the photos stored in the device's emulated storage (e.g., camera images and downloaded content). This artifact confirms that Google Photos was actively used to preview or sync media on the device, providing supplementary evidence of user interaction with images even if full-resolution photos are not directly stored in this cache.

ALEAPP

Google Photos (gphotos-1) - Cache report

Total number of entries: 9

Google Photos (gphotos-1) - Cache located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.photos\databases\disk_cache

Show 15 entries

Search:

Timestamp	Key	Image	Size	Pending Deletion
2025-11-25 06:00:44	2032a1531c6f4c653340deda0cbad7a89d33a00599437f01fdd4ae2eb76ee372		107315	
2025-11-25 06:00:44	2cd3634f46ad48216c9fe7f2549d1711c3118bed50212e6d7635c5b79fca44f		21398	
2025-11-25 06:00:49	65389ff6f97c291c32b14c002419f70638f71e2bbaa1c0c255d3cce58eed2e73		2748	
2025-11-25 06:00:44	81fbef6ff49994aaf662df6b316d8dfe8a3b56e3bc9ea160a19b8ebb5e4f05f60		4976	

3.3.10 Google Photos Local Media

The Google Photos *Local Media* database provides a clear record of media files stored on the device, including both camera-captured photos and downloaded images. Each entry contains rich metadata such as resolution, file size, folder location, and precise timestamps showing when the images were captured or downloaded. Notably, the database logs two camera photos taken moments apart, two cybersecurity-related images downloaded via Chrome, and one Instagram-related image saved to the Pictures/Instagram folder. This artifact also includes inferred GPS coordinates for the camera photos, allowing the reconstruction of the user's approximate location at the time of capture. Overall, this dataset confirms active camera use, browsing-driven downloads, and Instagram media storage, making it highly valuable for timeline building and activity correlation.

ALEAPP

Google Photos (gphotos-1) - Local Media report

Total number of entries: 5

Google Photos (gphotos-1) - Local Media located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.google.android.apps.photos\databases\gphotos-1.db

Show 15 entries

Search:

Timestamp	File Name	File Path	Captured Timestamp (Local)	Timezone Offset	Width	Height	Size	Duration	Latitude	Longitude	Folder Name	Media Store ID
2025-11-25 00:49:46	IMG_20251124_164946.jpg	/storage/emulated/0/Pictures/IMG_20251124_164946.jpg	2025-11-24 16:49:46	-8	1392	1856	37506		37.4234	-122.084	Pictures	17
2025-11-25 00:49:49	IMG_20251124_164949.jpg	/storage/emulated/0/Pictures/IMG_20251124_164949.jpg	2025-11-24 16:49:49	-8	1392	1856	38585		37.4234	-122.084	Pictures	18
2025-11-25 00:51:28	cybersecurity.png	/storage/emulated/0/Download/cybersecurity.png	2025-11-24 16:51:28	-8	320	200	47677				Download	19
2025-11-25 00:52:05	13_P-600x337.png	/storage/emulated/0/Download/13_P-600x337.png	2025-11-24 16:52:05	-8	600	337	328412				Download	21
2025-11-25 03:16:05	1764040565437.jpg	/storage/emulated/0/Pictures/Instagram/1764040565437.jpg	2025-11-24 19:16:05	-8	320	320	54597				Instagram	23
Timestamp	File Name	File Path	Captured Timestamp (Local)	Timezone Offset	Width	Height	Size	Duration	Latitude	Longitude	Folder Name	Media Store ID

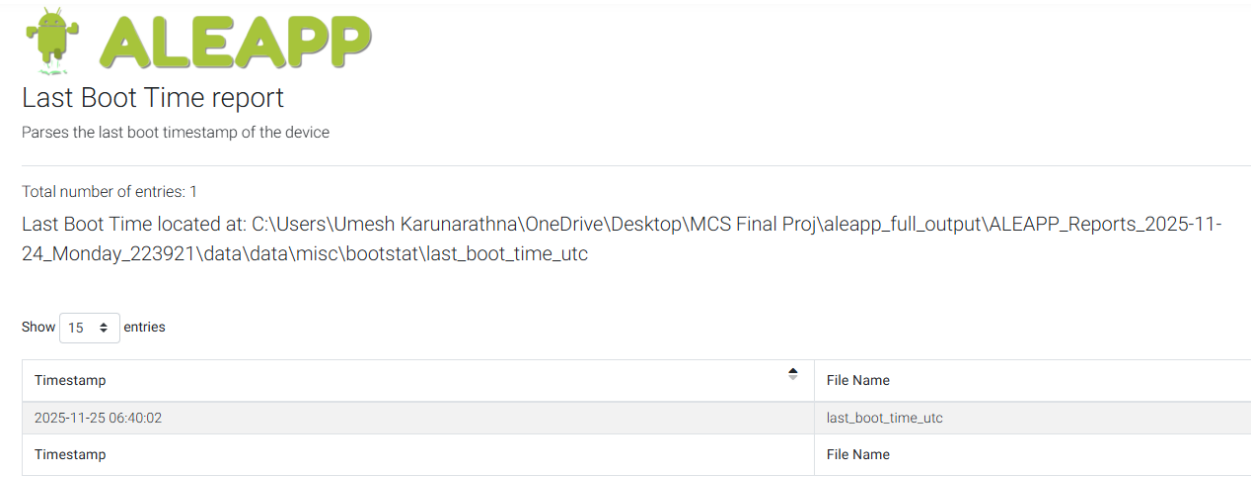
3.3.11 Installed Applications

The device contained a mix of system apps and user-installed applications, identified through multiple sources including the Nexus Launcher icon database, Google Play link mappings, and the gass.db installed-apps table. Core system apps such as **Camera**, **Chrome**, **Contacts**, **Gmail**, **Google**, **Maps**, **Messages**, and **Settings** were present, confirming a standard Android setup. User-installed apps included: **Dropbox**, **Instagram Lite**, **Instagram (full app)**, **Pinterest**, **YouTube Music**, and remnants of **TikTok (com.zhiliaoapp.musically)**. Each entry in the Installed Apps report provided bundle IDs, version codes, and SHA-256 hashes, supporting app integrity verification and version attribution.

The App Icons report further validated these installations by listing launcher-visible apps and their shortcuts, providing an additional layer of confirmation. Meanwhile, the Google Play Links report enumerated all packages on the device and mapped them to possible Play Store URLs, including system components, which helps distinguish between user-installed and pre-loaded services. The full packages.xml record listed over 200 packages, including system modules, APEX components, and Google services, along with their install/update timestamps. Together, these artifacts provide a complete view of the applications installed on the device and help correlate app activity with other forensic findings.

3.3.12 Last Boot Time

The device's last reboot timestamp was extracted from the bootstat artifact located at data/misc/bootstat/last_boot_time_utc. The report shows a single entry indicating the device was last booted on **2025-11-25 at 06:40:02 UTC**. This timestamp is useful for correlating user activity, validating shutdown/startup sequences observed in Digital Wellbeing logs, and confirming that all captured artifacts reflect activity occurring after the most recent system start-up.



3.3.13 SMS Messages

The SMS database (mmssms.db) from **com.android.providers.telephony** contains two outgoing messages sent from the device. Both messages were sent on **2025-11-25** to contacts already present in the device's contact list. The first message ("Not urgent, we

can chat later.”) was sent to **544258844**, and the second message (“Hello shane”) was sent to **455566774**. Both are marked as **sent and read**, indicating successful delivery with no errors. These records help reconstruct short-form communication activity and show that the device was actively used for messaging shortly before the data extraction.

ALEAPP

SMS messages report

Total number of entries: 2

SMS messages located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\data\com.android.providers.telephony\databases\mmssms.db

Show 15 entries

Search:

Date	MSG ID	Thread ID	Address	Contact ID	Date sent	Read	Type	Body	Service Center	Error code
2025-11-25 05:59:49	1	3	544258844			1	Sent	Not urgent, we can chat later.		-1
2025-11-25 06:02:16	2	4	455566774			1	Sent	Hello shane		-1
Date	MSG ID	Thread ID	Address	Contact ID	Date sent	Read	Type	Body	Service Center	Error code

3.3.14 Wi-Fi Hotspot Configuration

The **Wi-Fi Hotspot report** extracted from WifiConfigStoreSoftAp.xml reveals that the device had an active mobile hotspot configuration saved. The hotspot SSID is **“AndroidAP_3257”**, and ALEAPP successfully extracted the **passphrase**, which is shown in plaintext as **yr4ppff36wv2q5q**. The security type value (**1**) corresponds to **WPA2-PSK**, the standard security protocol for Android hotspots.

This artifact is highly relevant because it demonstrates:

- The device had hotspot functionality configured.
- The exact hotspot name and password were stored in system configuration files.
- Sensitive network credentials can be recovered from rooted or emulated devices.
- This could help investigators determine whether the device was used to provide tethering to other devices.

ALEAPP

Wi-Fi Hotspot report

Total number of entries: 1

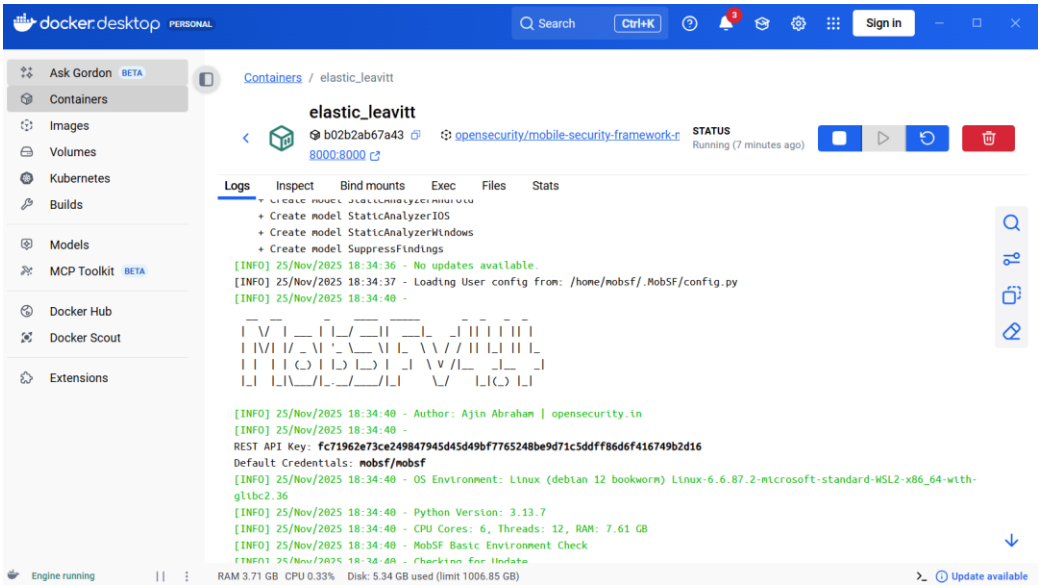
Wi-Fi Hotspot located at: C:\Users\Umesh Karunarathna\OneDrive\Desktop\MCS Final Proj\aleapp_full_output\ALEAPP_Reports_2025-11-24_Monday_223921\data\data\misc\apexdata\com.android.wifi\WifiConfigStoreSoftAp.xml

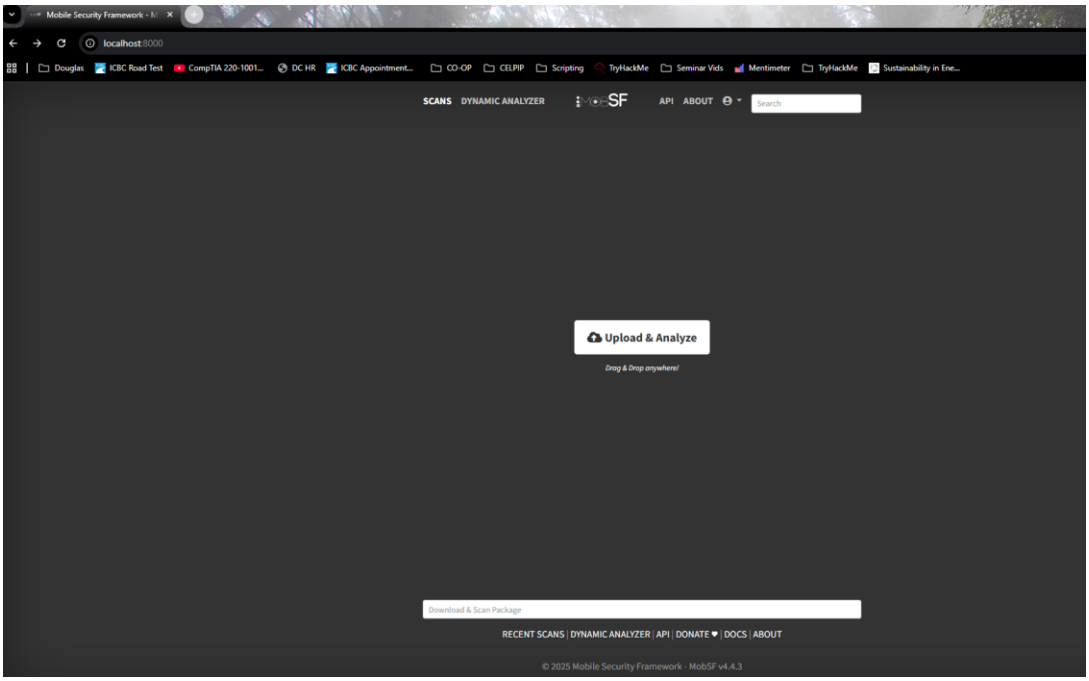
Show 15 entries

SSID	Passphrase	SecurityType
AndroidAP_3257	yr4ppff36wv2q5q	1
SSID	Passphrase	SecurityType

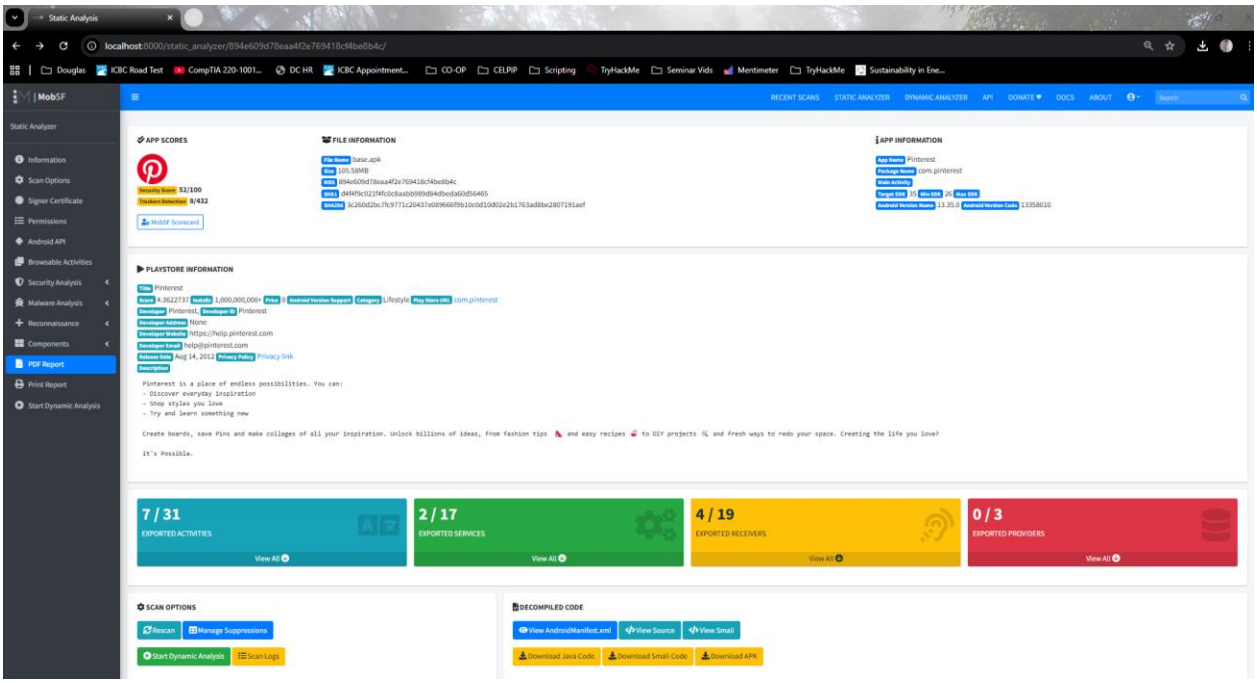
3.4 MobSF

Mobile Security Framework (MobSF) was set up using its Docker-based deployment, which is the recommended method because it automatically configures all required dependencies and services. Once the MobSF container was running, the APKs were uploaded to perform **static analysis**, which examines components such as manifests, permissions, certificates, activities, services, receivers, providers, and possible hardcoded secrets. MobSF also supports **dynamic analysis** through an instrumented emulator to capture runtime behaviors, API calls, network traffic, and file system changes, although only static analysis was required for this project. The Docker setup ensured a stable, isolated environment for accurate security and forensic assessment of Android applications.





Pinterest APK – Static Analysis using MobSF



The Pinterest APK was successfully uploaded into MobSF and fully processed using the Static Analyzer. MobSF extracted the AndroidManifest.xml, revealing a highly permission-rich application with extensive integrations, third-party SDKs, and background services. Pinterest requests access to internet, network state, camera, audio recording, contacts, accounts, location, notifications, and media storage, indicating broad functionality and high data access potential. The manifest also contains multiple deep links, custom tab handlers, Firebase messaging services, Facebook SSO components, Google Maps API keys, and widget providers showing Pinterest's wide interaction range with both system services and external apps.

Pinterest – Manifest Analysis using MobSF

From a forensic perspective, the manifest clearly identifies multiple important entry points (activities), background processes (services), and broadcast receivers that can store logs, authentication tokens, and cache data. Pinterest also uses Room databases, WorkManager jobs, and a large number of custom content providers highlighting potential locations within /data/data/com.pinterest/ where user-related or operational artifacts may be found. This makes Pinterest a stronger forensic target than Instagram Lite, with richer local storage patterns and more identifiable components.

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="13358010" android:versionName="13.35.0" android:installLocation="auto" android:compileSdkVersion="35" android:compileSdkVersionCodename="15" package="com.pinterest" platformBuildVersionCode="35" platformBuildVersionName="15"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="26" android:targetSdkVersion="35" />
    <uses-permission android:name="com.pinterest.account.Credentials" />
    <permission android:name="com.pinterest.account.Credentials" android:protectionLevel="signature" />
    <uses-feature android:glEsVersion="0x00030000" android:required="true" />
    <supports-screens android:smallScreens="true" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" android:maxSdkVersion="32" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" android:maxSdkVersion="29" />
    <uses-permission android:name="android.permission.READ_MEDIA_IMAGES" />
    <uses-permission android:name="android.permission.READ_MEDIA_VIDEO" />
    <uses-permission android:name="android.permission.READ_MEDIA_VISUAL_USER_SELECTED" />
    <uses-permission android:name="com.google.android.gms.permission.AD_ID" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="com.samsung.android.mapsagent.permission.READ_APP_INFO" />
    <uses-permission-sdk-23 android:name="android.permission.SET_WALLPAPER" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission-sdk-23 android:name="android.permission.RECORD_AUDIO" />
    <uses-feature android:name="android.hardware.camera.autofocus" android:required="false" />
    <uses-feature android:name="android.hardware.camera.front" android:required="false" />
    <uses-feature android:name="android.hardware.camera.flash" android:required="false" />
    <uses-feature android:name="android.hardware.telephony" android:required="false" />
    <uses-feature android:name="android.hardware.location.gps" android:required="false" />
    <uses-feature android:name="android.hardware.nfc" android:required="false" />
    <uses-permission android:name="com.sec.android.provider.badge.permission.READ" />
    <uses-permission android:name="com.sec.android.provider.badge.permission.WRITE" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="android.permission.NFC" />
    <uses-feature android:name="android.hardware.camera" android:required="false" />
    <queries>
        <package android:name="com.pinterest.shuffles" />
        <package android:name="com.pinterest.shuffles.dev" />
        <package android:name="com.android.vending" />
        <package android:name="com.google.android.engage.verifyapp" />
        <intent>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.BROWSABLE" />
            <data android:scheme="https" />
        </intent>
        <intent>
            <action android:name="android.support.customtabs.action.CustomTabsService" />
        </intent>
        <package android:name="jp.naver.line.android" />
        <package android:name="com.facebook.katana" />
        <intent>
            <action android:name="com.appsflyer.referrer.INSTALL_PROVIDER" />
        </intent>
        <package android:name="com.instagram.android" />
        <package android:name="com.facebook.lite" />
        <package android:name="com.samsung.android.mapsagent" />
    </queries>
    <queries>
        <package android:name="com.samsung.android.mapsagent" />
        <package android:name="com.whatsapp" />
        <package android:name="com.facebook.katana" />
        <package android:name="com.facebook.orca" />
        <package android:name="com.facebook.lite" />
        <package android:name="com.facebook.mlite" />
        <package android:name="com.twitter.android" />
        <package android:name="com.tencent.mm" />
```

Pinterest – Permission Analysis using MobSF

MobSF reveals that Pinterest requests a wide range of dangerous permissions, indicating the app's deep interaction with device resources and user data. These include location permissions (coarse and fine), which allow Pinterest to pinpoint the user's physical location, potentially for targeted content, ads, and tracking. Camera and audio permissions enable capturing photos, videos, and possibly recording in-app content. Permissions to read contacts and accounts give Pinterest visibility into account identities stored on the device, while media-read permissions provide broad access to photos and videos stored on the device or selected via the photo picker. Combined, these capabilities allow Pinterest to ingest user-generated content, profile data, and contextual metadata.

From a forensic standpoint, these permissions highlight what data categories Pinterest is allowed to access and therefore which artifacts may appear in /data/data/com.pinterest/. Access to external storage and media means thumbnails, cached images, and temporary uploads can appear in the app's cache directories. Location permissions can result in geotagged content or location-based API calls stored in logs. Access to accounts, contacts, and notifications suggests the possibility of tokens, identifiers, and message metadata being saved locally. Overall, the requested permissions indicate that Pinterest performs extensive background operations and content processing, making it a high-value target for forensic analysis.

APPLICATION PERMISSIONS

Search:

PERMISSION	STATUS	INFO	DESCRIPTION	CODE MAPPINGS
android.permission.ACCESS_COARSE_LOCATION	dangerous	coarse (network-based) location	Access coarse location sources, such as the mobile network database, to determine an approximate phone location, where available. Malicious applications can use this to determine approximately where you are.	Show Files
android.permission.ACCESS_FINE_LOCATION	dangerous	fine (GPS) location	Access fine location sources, such as the Global Positioning System on the phone, where available. Malicious applications can use this to determine where you are and may consume additional battery power.	Show Files
android.permission.CAMERA	dangerous	take pictures and videos	Allows application to take pictures and videos with the camera. This allows the application to collect images that the camera is seeing at any time.	Show Files
android.permission.GET_ACCOUNTS	dangerous	list accounts	Allows access to the list of accounts in the Accounts Service.	Show Files
android.permission.POST_NOTIFICATIONS	dangerous	allows an app to post notifications.	Allows an app to post notifications	Show Files
android.permission.READ_CONTACTS	dangerous	read contact data	Allows an application to read all of the contact (address) data stored on your phone. Malicious applications can use this to send your data to other people.	Show Files
android.permission.READ_EXTERNAL_STORAGE	dangerous	read external storage contents	Allows an application to read from external storage.	Show Files
android.permission.READ_MEDIA_IMAGES	dangerous	allows reading image files from external storage.	Allows an application to read image files from external storage.	Show Files
android.permission.READ_MEDIA_VIDEO	dangerous	allows reading video files from external storage.	Allows an application to read video files from external storage.	Show Files
android.permission.READ_MEDIA_VISUAL_USER_SELECTED	dangerous	allows reading user-selected image or video files from external storage.	Allows an application to read image or video files from external storage that a user has selected via the permission prompt photo picker. Apps can check this permission to verify that a user has decided to use the photo picker, instead of granting access to READ_MEDIA_IMAGES or READ_MEDIA_VIDEO . It does not prevent apps from accessing the standard photo picker manually. This permission should be requested alongside READ_MEDIA_IMAGES and/or READ_MEDIA_VIDEO , depending on which type of media is desired.	

Showing 1 to 10 of 31 entries

Previous 1 2 3 4 Next

Possible Hardcoded Secrets of Pinterest APK using MobSF

MobSF detected a large number of strings under *Possible Hardcoded Secrets*, but most of these entries are localization strings such as translations for “password,” “secret,” and “cancel” in multiple languages. These are not real credentials they are simply UI strings for app menus, privacy options, and authentication prompts.

However, a few entries are true hardcoded values and have forensic relevance. The manifest exposes a Google API key, a Firebase database URL, and a Facebook Client Token, which indicate the app’s connection endpoints and third-party SDK integrations. These values do not directly reveal user data, but they show which external services Pinterest communicates with and could help identify network traffic, backend endpoints, or Firebase-linked metadata during deeper forensic analysis. In summary, most items are harmless translations, but the presence of API keys confirms the app relies on cloud-connected services that may store or process user information remotely.

POSSIBLE HARDCODED SECRETS

▼ Show all **952** secrets

"android.credentials.TYPE_PASSWORD_CREDENTIAL" : "Adgangskode"
"your_private_profile_popup_body" : "あなたのプロフィール、ピン、ボード、フォロワー、フォロー中のリストを閲覧できるのは、承認されたフォロワーのみです。"
"android.credentials.TYPE_PASSWORD_CREDENTIAL" : "Пароль"
"settings_security_connected_devices_revoke_session_cancel" : "Otkaži"
"androidx.credentials.TYPE_PUBLIC_KEY_CREDENTIAL" : "Klucz"
"gbl_password" : "密碼"
"api_latest_host_uri" : "api-latest.pinterest.com"
"settings_security_connected_devices_revoke_session_cancel" : "Avbryt"
"filter_secret" : "Verborgen"
"your_private_profile_popup_title" : "個人檔案未公開"
"settings_security_connected_devices_revoke_session_cancel" : "Cancel"
"add_your_password" : "パスワードを入力してください"
"android.credentials.TYPE_PASSWORD_CREDENTIAL" : "Пароль"
"secret_board_label" : "隱私"
"missing_username" : "ユーザー名が入力されていません"
"accessibility_show_password" : "パスワードを表示"
"filter_secret" : "Tajni"
"password" : "Parola"
"firebase_database_url" : "https://pinterest-android.firebaseio.com"
"gbl_password" : "Lozinka"
"unfollow_user" : "Ontvolg"
"add_your_password" : "添加你的密碼"
"gbl_password" : "密碼"
"gbl_password" : "ةمكلا"
"private_policy" : "Privaatheidsbeleid"
"filter_secret" : "Тайни"
"unfollow_user" : "เลิกติดตาม"
"unfollow_user" : "Leiratkozás"
"filter_secret" : "私密"
"settings_account_management_password" : "Password"
"show_password" : "แสดงรหัสผ่าน"
"filter_secret" : "ئىمە"
"filter_secret" : "シークレット"
"settings_account_management_password" : "Wachtwoord"
"secret_board" : "隱私圖版"
"filter_secret" : "ΠΙΟ"
"unfollow_pin_user" : "「%1$s」のピンを表示しない"
"password" : "密碼"
"google_api_key" : "AIzaSyDybBm_hqwipeuJdUDfpMjYdPxGG3jJjAA"
"android.credentials.TYPE_PASSWORD_CREDENTIAL" : "密碼"
"unfollow_user" : "Dezabonare"
"settings_private_profile_restricted_setting_education" : "出于隐私原因，此设置不可用"
"settings_security_connected_devices_revoke_session_cancel" : "Anuluj"
"filter_secret" : "Salainen"
"gbl_password" : "パスワード"
"gbl_password" : "รหัสผ่าน"

Network Security Results of Pinterest APK using MobSF

MobSF's network security analysis shows that the app's base security configuration is weak, because the global network settings allow clear-text (HTTP) traffic to *any* domain. This is marked as a high-severity issue since it could expose sensitive data if transmitted without encryption. However, for major production domains such as pinterest.com, pinimg.com, branch.io, facebook.com, appsflyer.com, bugsnag.com, and others the app correctly enforces HTTPS-only communication, meaning clear-text traffic is explicitly blocked for these destinations. This indicates that while the broader configuration is permissive, the app's essential services and SDK integrations still communicate securely using encrypted channels.

NETWORK SECURITY

HIGH	WARNING	INFO	SECURE
1	0	0	1

NO	SCOPE	SEVERITY	DESCRIPTION
1	*	high	Base config is insecurely configured to permit clear text traffic to all domains.
2	pinterest.com pinimg.com branch.io facebook.com appsflyer.com bugsnag.com cedexis.com	secure	Domain config is securely configured to disallow clear text traffic to these domains in scope.

Signer Certificate Analysis using MobSF

The APK is signed with an official Pinterest Inc. X.509 certificate, confirming the binary is legitimate and originates from the publisher. The app uses v2 and v3 signing schemes, which protect against tampering on modern Android versions, while v1 and v4 signatures are not present. The certificate is long-lived (valid from 2012 to 2037) and self-issued by Pinterest, which is normal for production apps. However, the public key size is 1024-bit RSA, which is considered weak by today's security standards, even though it is still commonly found in older Android signing infrastructures. Overall, the signature data confirms authenticity but highlights that the cryptographic strength could be improved by moving to a stronger key (2048-bit or above) for future releases.

SIGNER CERTIFICATE

```
Binary is signed
v1 signature: False
v2 signature: True
v3 signature: True
v4 signature: False
X.509 Subject: C=US, ST=CA, L=Palo Alto, O=Pinterest Inc, OU=Android, CN=Carl Rice
Signature Algorithm: rsassa_pkcs1v15
Valid From: 2012-04-24 16:30:16+00:00
Valid To: 2037-04-18 16:30:16+00:00
Issuer: C=US, ST=CA, L=Palo Alto, O=Pinterest Inc, OU=Android, CN=Carl Rice
Serial Number: 0x4f96d518
Hash Algorithm: sha1
md5: 418c57c22436e3297b8a837edc9882bd
sha1: b6a74dbcb894b0f73d8c485c72eb1247a8f027ca
sha256: 341d6881b1ecf38361fbf8c8fbae0aa516b45375c39ef5e78b161869acc1bcfa
sha512: 902c900f3a5aa005f682cfb5cdd18f28d01e2026fc2f34453b16b8342ab48cef1cf1dfcd152feaf98df96d4dcdab71a6155da52eeae636f2196ba60f619c265a
PublicKey Algorithm: rsa
Bit Size: 1024
Fingerprint: 0b97405d1097740bed2ad1677dfcfd25ce288e0419991aa973d171c4f2a24a76
Found 1 unique certificates
```

Component Analysis using MobSF

The MobSF static analysis of the Pinterest APK reveals a large and complex component structure, indicating a feature-rich application with extensive background processing, user interaction flows, and third-party integrations. The app includes **31 Activities**, covering major user functions such as authentication, media capture, profile management, comments, WebView rendering, Pinterest board creation, and Share/Repin workflows. These show how the app handles UI navigation and user-initiated tasks.

The application also registers **17 background Services**, including Firebase messaging, WorkManager job services, the Pinterest PinUploaderService, MQTT-based messaging, and several components handling authentication, delayed startup, board collaboration, and multi-process invalidation. These services confirm that the app performs continuous background sync, push handling, and scheduled tasks even when not in active use highly relevant for forensic timelines.

Additionally, the app exposes **19 Broadcast Receivers**, including Facebook token receivers, device boot listeners, WorkManager proxy receivers, network state watchers, and custom Pinterest widgets. These receivers allow the app to react to system-level triggers such as network changes, boot events, power state changes, and external shares providing more forensic points for activity reconstruction.

The APK also contains **3 Content Providers**, including the Pinterest CredentialsContentProvider and a FileProvider, both important because they influence how the app stores and shares files or credential-related metadata. The presence of **four embedded libraries** (Apache HTTP legacy, Window Extensions, Sidecar, and AdServices) shows reliance on system frameworks and backward-compatibility modules.

ACTIVITIES

▼ Showing all **31** activities

com.pinterest.activity.PinterestActivity
com.pinterest.activityLibrary.activity.task.activity.MainActivity
com.pinterest.activity.ExperimentsReloaderActivity
com.facebook.FacebookActivity
com.pinterest.widget.configuration.WidgetConfigurationActivity
com.pinterest.feature.unifiedcomments.view.CommentActivity
com.pinterest.componentBrowser.ComponentBrowserActivity
com.pinterest.activity.webhook.WebhookActivity
com.pinterest.identity.UnauthActivity
com.pinterest.account.AuthenticatorActivity
com.pinterest.activity.web.WebViewActivity
com.pinterest.feature.mediagallery.MediaGalleryActivity
com.pinterest.activity.NavBaseActivity
com.pinterest.activity.nux.NUXActivity
com.pinterest.feature.pin.creation.CreationActivity
com.pinterest.activity.create.CameraActivity
com.pinterest.activity.create.PinItActivity
com.pinterest.activity.create.ScrapedImagesResultsActivity
com.pinterest.activity.user.UserSetImageActivity
com.pinterest.activity.create.RepinActivity
com.pinterest.activity.SendShareActivity
com.pinterest.unauth.controller.SSOActivity
com.pinterest.library.navigation.activity.NavActivity
com.linecorp.linesdk.auth.internal.LineAuthenticationActivity
com.linecorp.linesdk.auth.internal.LineAuthenticationCallbackActivity
com.facebook.CustomTabMainActivity
com.facebook.CustomTabActivity
androidx.credentials.playservices.HiddenActivity
com.google.android.gms.auth.api.signin.internal.SignInHubActivity
com.google.android.gms.common.api.GoogleApiActivity
com.google.android.play.core.common.PlayCoreDialogWrapperActivity

SERVICES

▼ Showing all **17** services

info.mqtt.android.service.MqttService
com.pinterest.account.AuthenticatorService
com.pinterest.service.DelayedStartupService
com.pinterest.service.PinUploaderService
com.pinterest.pushnotification.MessagingService
com.pinterest.feature.board.collab.service.BoardActionService
androidx.work.impl.background.systemalarm.SystemAlarmService
androidx.work.impl.background.systemjob.SystemJobService
androidx.work.impl.foreground.SystemForegroundService
androidx.camera.core.impl.MetadataHolderService
com.google.firebase.messaging.FirebaseMessagingService
com.google.firebase.components.ComponentDiscoveryService
androidx.credentials.playservices.CredentialProviderMetadataHolder
com.google.android.gms.auth.api.signin.RevocationBoundService
com.google.android.gms.measurement.AppMeasurementService
com.google.android.gms.measurement.AppMeasurementJobService
androidx.room.MultiInstanceInvalidationService

RECEIVERS

▼ Showing all **19** receivers

androidx.profileinstaller.ProfileInstallReceiver
com.pinterest.identity.core.controller.facebook.broadcast.FacebookBroadcastReceiverWrapper$AccessTokenChangedReceiver
com.pinterest.identity.core.controller.facebook.broadcast.FacebookBroadcastReceiverWrapper$AccessTokenExpiredReceiver
com.pinterest.account.AccountTransferBroadcastReceiver
com.pinterest.browser.customTabs.chrome.ChromeTabBroadcastReceiver
com.pinterest.feature.pear.util.ApplicationSelectionReceiver
com.pinterest.social.ExternalShareSelectionBroadcastReceiver
com.pinterest.widget.GalleryWallWidget
com.pinterest.widget.SingleImageWidget
com.pinterest.engage.GoogleEngageBroadcastReceiver
androidx.work.impl.utils.ForceStopRunnable$BroadcastReceiver
androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryChargingProxy
androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryNotLowProxy
androidx.work.impl.background.systemalarm.ConstraintProxy$StorageNotLowProxy
androidx.work.impl.background.systemalarm.ConstraintProxy$NetworkStateProxy
androidx.work.impl.background.systemalarm.RescheduleReceiver
androidx.work.impl.background.systemalarm.ConstraintProxyUpdateReceiver
com.google.firebase.iid.FirebaseInstanceIdReceiver
com.google.android.gms.measurement.AppMeasurementReceiver

PROVIDERS

▼ Showing all **3** providers

com.pinterest.account.CredentialsContentProvider
androidx.startup.InitializationProvider
androidx.core.content.FileProvider

LIBRARIES

▼ Showing all **4** libraries

org.apache.http.legacy
androidx.window.extensions
androidx.window.sidecar
android.ext.adservices

3.5 Strava

During the initial phase, Strava launched correctly on the emulator and allowed me to log in using the test account. At that point, I was briefly able to view basic profile information such as **height, weight, gender, and account metadata**, confirming that the login process and initial sync were working. However, after a few attempts, the app repeatedly failed to authenticate, and Strava eventually became completely inaccessible inside the emulator.


```
)
c{"activity_visibility":"everyone","all_time_activity_count":0,"athlete_type":1,"autoplay_video":true,"badge_type_id":0,"chat_invite_visibility":"followers_only","city":"Vancouver","club
s":[],"connected_devices":[],"consents":{"aggregate_data":"unknown","privacy_policy":"approved","terms_of_service":"approved","health":"unknown","location":"unknown","direct_promotion":"
approved","age_confirmation":"approved"},"contact_sync_enabled":true,"country":"Canada","created_at":"2025-10-30T04:42:08Z","dateofbirth":"1997-11-16","default_photo_enabled":true,"descr
iption":"","email":"csis4440user@gmail.com","email_verification_status":"verified","external_identities":[{"identity_hash":"d4b3902d8496057cf388a823cf4c5ec61a88aa264678ae478cc0e783ed9b9c
36","identity_name":"BRANCH"}],"firstname":"CSIS440","flyby_visibility":"no_one","friend_count":3,"gender":"man","group_activity_visibility":"everyone","has_password":false,"has_trial_of
fer":false,"id":192356605,"is_trial_eligible":true,"is_winback_via_view":true,"lastname":"User","local_legend_visibility":"everyone","map_vis_local_global_experiment_eligible":false,"max
_heartrate":193,"measurement_preference":"meters","mentions_visibility":"everyone","metro_heatmap_visibility":"everyone","partner_opt_outs":[],"plan":"free","postable_clubs_count":0,"pre
mium":false,"prioritize_recent_activities_feed":false,"profile":"https://lh3.googleusercontent.com/a/ACg8ocLRyMgvLtNm3MkyPi7kCOaADun9VYojvks-3T6vuy87IcXkhw\u003ds96-
c","profile_visibility":"everyone","profile_medium":"https://lh3.googleusercontent.com/a/ACg8ocLRyMgvLtNm3MkyPi7kCOaADun9VYojvks-3T6vuy87IcXkhw\u003ds96-
c","profile_original":"https://lh3.googleusercontent.com/a/ACg8ocLRyMgvLtNm3MkyPi7kCOaADun9VYojvks-3T6vuy87IcXkhw\u003ds96-
c","recurring":"Monthly","sample_race_distance":0,"sample_race_time":0,"sex":"M","signup_name_required":false,"state":"British
Columbia","tour_de_france_opt_in":"opted_out","username":"csis440_user","weight":63.0}
{"athlete_id":192356605,"is_premium":false,"is_trial_eligible":true,"state":"FREE","sub_state":"EMPTY"}
+?{"athlete_id":192356605,"is_premium":false,"is_trial_eligible":true,"state":"FREE","sub_state":"EMPTY"}
Nl{"athlete_id":192356605,"is_premium":false,"is_trial_eligible":true,"state":"FREE","sub_state":"EMPTY"}
```


Below are the consolidated issues that prevented further forensic analysis:

1. Initial Login Worked Once, Then Completely Failed

Strava successfully logged in only during the first attempt. On subsequent runs:

- Login failed repeatedly despite correct credentials
- Password reset and device unlinking did not help
- Clearing app data, reinstalling APKs, and creating new emulator instances also failed

This made it impossible to generate or re-sync activities, preventing the app from producing forensic artifacts.

2. Activity Sync Never Appeared on Emulator

Even when real activities were created on a physical device:

- They did not sync into the emulator's Strava installation
- Database tables remained empty
- No activity caches, map tiles, or FIT files were generated

This indicated that the emulator Strava instance could not communicate with Strava's backend reliably.

3. Native Library Crash: libFatmapSdk.so

Attempts to manually inject activity JSON into the database caused a fatal crash:

libFatmapSdk.so not found

This library is required for Strava's 3D/2D map rendering. Without it, the app cannot load or display activities, resulting in zero artifact generation.

4. Permission Errors During Extraction

While extracting /data/data/com.strava/, some directories—especially:

- .fit
- .mapbox
- files/

returned *permission denied* even with ADB root. These locations typically hold the most valuable forensic evidence (cached tiles, offline maps, raw GPS traces).

5. ALEAPP Returned No Usable Artifacts

After running ALEAPP on the extracted folders:

- Reports were empty or contained only structural metadata
- No activities, sync timestamps, or media files were detected

This confirmed that the app never created the required artifact footprint.

6. No Meaningful Data for Timeline or Correlation

Because Strava could not produce:

- Activity logs
- GPS traces
- Cached thumbnails
- User movement timeline
- Sync logs

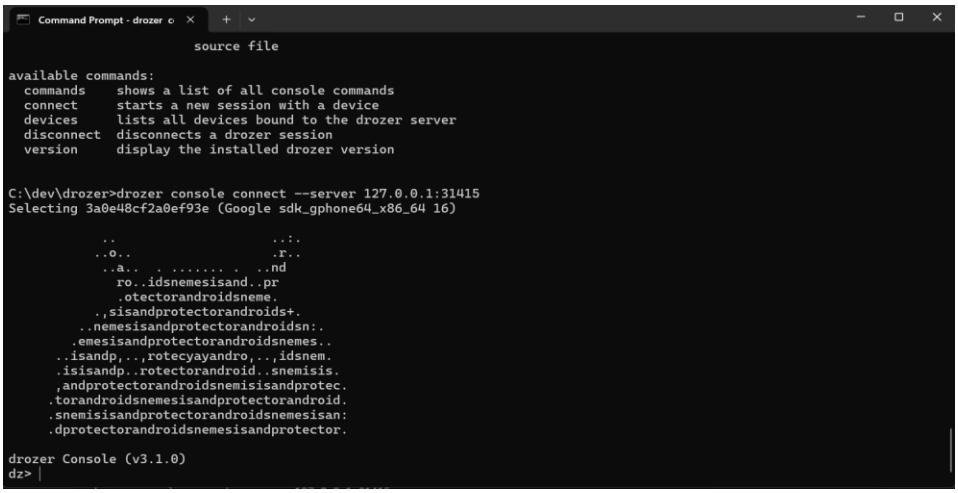
there was no usable evidence to correlate with Dropbox or other apps

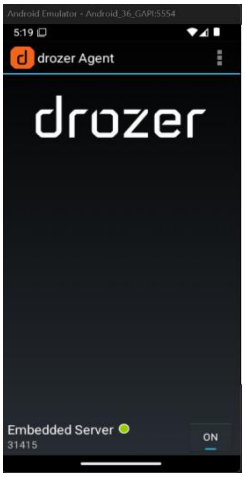
3.6 Drozer

Drozer is an Android security assessment framework that runs an agent on the device and provides a console interface from the attacker's point of view. Instead of accessing files directly, Drozer operates at the **inter-process communication (IPC)** level: it interacts with Activities, Services, Broadcast Receivers, and Content Providers using the same permissions as a normal third-party app.

This makes Drozer ideal for:

- Enumerating exported app components.
- Mapping an application's attack surface.
- Querying content providers to reveal data structures.
- Launching internal Activities without using the app's normal UI flow.
- Detecting misconfigured or vulnerable components.





3.7 Spotify Analysis using Drozer

Attack Surface Mapping

The first Drozer target was Spotify (com.spotify.music). I used the following command to map its attack surface:

run app.package.attacksurface com.spotify.music

This command shows the count of exported Activities, Services, Receivers, and Providers that can be reached from another app.

```
dz> run app.package.attacksurface com.spotify.music
Attempting to run shell module
Attack Surface:
  8 activities exported
  11 broadcast receivers exported
  3 content providers exported
  8 services exported
dz>
```

From this, it is clear that Spotify exposes a limited set of IPC entry points. Most components are not exported, which is what we would expect from a hardened production app.

Content Providers and Schema Enumeration

Spotify's content providers:

run app.provider.info -a *com.spotify.music*

```
dz> run app.activity.info -a com.spotify.music
Attempting to run shell module
Package: com.spotify.music
  com.spotify.music.MainActivity
    Permission: null
  com.spotify.music.features.waze.WazeReturnActivity
    Permission: null
  com.spotify.music.features.partneraccountlinking.PartnerAccountLinkingActivity
    Permission: null
  com.spotify.login.loginflowimpl.QuickLoginActivity
    Permission: null
  com.spotify.mobile.android.sso.AuthorizationActivity
    Permission: null
  com.spotify.music.externallogin.LoginRedirectActivity
    Permission: null
  com.spotify.login.magiclink.MagicLinkActivity
    Permission: null
  com.facebook.CustomTabActivity
    Permission: null

dz> |
```

Drozer reported multiple content providers, including providers related to messaging and device connectivity (Spotify Connect). To understand what these providers expose, I queried them directly:

run app.provider.query content://com.spotify.music.messaging.provider

run app.provider.query content://com.spotify.music.connect.provider

```
dz> run app.provider.query content://com.spotify.music.connect.provider
Attempting to run shell module
| _id | device_name | device_type | device_tech | device_state | disabled_reason | is_self | social_session |

dz> run app.provider.query content://com.spotify.music.messaging.provider
Attempting to run shell module
| _id | body | positive_action | negative_action |
```

Even though Spotify does not leak any actual user data inside my environment, Drozer was able to enumerate the **schema columns** of these providers. For example, columns such as body, positive_action, negative_action appeared for the messaging provider,

and device_name, device_type, device_state, and other fields appeared for the connect provider.

This is not necessarily a vulnerability, but it is an important **reconnaissance insight**:

- A normal, unprivileged app can still learn about internal data structures of Spotify via content providers.
- An attacker could use this information to design fuzzing or injection attempts against these endpoints.
- From a forensic perspective, this schema helps interpret internal SQLite databases if they are acquired through other means.

Spotify therefore presents a **controlled but discoverable attack surface**.

3.8 Trust Wallet Analysis using Drozer

Trust Wallet is a cryptocurrency wallet application, which should have a much stricter security posture than typical consumer apps.

Attack Surface View

Using Drozer, I examined the attack surface for Trust Wallet (actual package name may vary depending on version):

run app.package.attacksurface com.wallet.crypto.trustapp

```
dz> run app.package.attacksurface com.wallet.crypto.trustapp
Attempting to run shell module
Attack Surface:
  1 activities exported
  1 broadcast receivers exported
  0 content providers exported
  0 services exported
dz>
```

Compared to Spotify, the Trust Wallet attack surface was significantly smaller. There were few, if any, exported activities or content providers that looked accessible without permissions. This matches expectations for a security-critical wallet app the less exported surface, the fewer places a malicious app can interact with it directly.

Provider / Debug Posture

I also checked whether Trust Wallet exposed content providers or debugging flags:

run app.provider.info -a com.wallet.crypto.trustapp

run app.package.debuggable com.wallet.crypto.trustapp

```
dz> run app.provider.info -a com.wallet.crypto.trustapp
Attempting to run shell module
Package: com.wallet.crypto.trustapp
  No matching providers.
```

```
dz> run app.package.debuggable -a com.wallet.crypto.trustapp
Attempting to run shell module
Exception occured: unrecognized arguments: -a com.wallet.crypto.trustapp
dz>
```

In my environment, no obviously sensitive providers were accessible, and the application was not debuggable. This means:

- Tools or apps cannot simply attach as the app's user using run-as.
- Direct content-provider abuse is limited.

From Drozer's point of view, Trust Wallet exposes **very few external hooks**, which is consistent with a hardened design for handling private keys and transaction signing.

3.9 InsecureBankv2 (Vulnerable App) Analysis using Drozer

To demonstrate Drozer's full capability to exploit misconfigurations, I analyzed **InsecureBankv2**, a deliberately vulnerable training application.

Exported Activities

First, I listed the activities:

run app.activity.info -a com.android.insecurebankv2

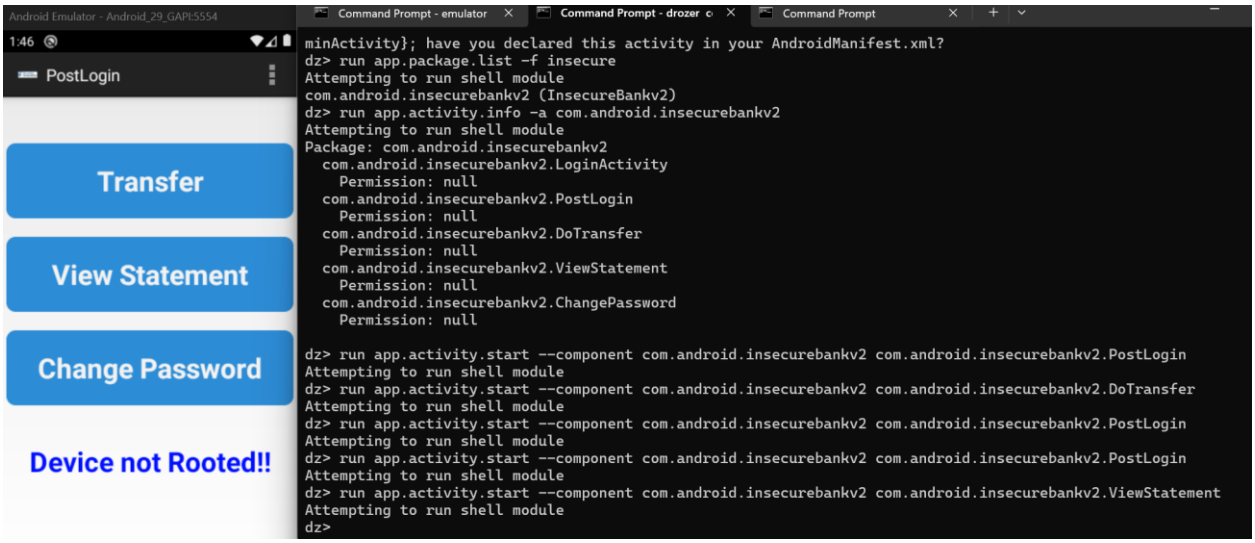
```
dz> run app.package.attacksurface com.android.insecurebankv2
Attempting to run shell module
Attack Surface:
  5 activities exported
  1 broadcast receivers exported
  1 content providers exported
  0 services exported
    is debuggable
dz>
```

Drozer returned multiple activities such as:

- com.android.insecurebankv2.LoginActivity
- com.android.insecurebankv2.PostLogin
- com.android.insecurebankv2.DoTransfer
- com.android.insecurebankv2.ViewStatement
- com.android.insecurebankv2.ChangePassword

Each of these had:

Permission: null

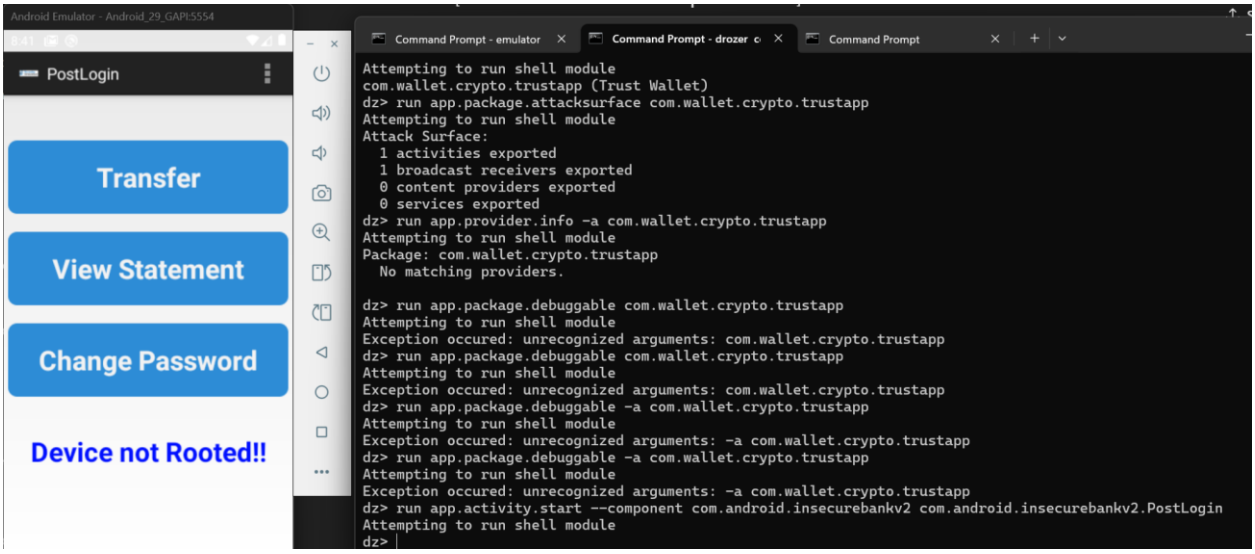


This means **any app (or Drozer) can launch them directly without any permission check**.

Login Bypass Using Drozer

In the normal app flow, a user must log in via LoginActivity before accessing the other screens. Using Drozer, I bypassed this officially required step by starting the PostLogin activity directly:

run app.activity.start --component com.android.insecurebankv2 com.android.insecurebankv2.PostLogin

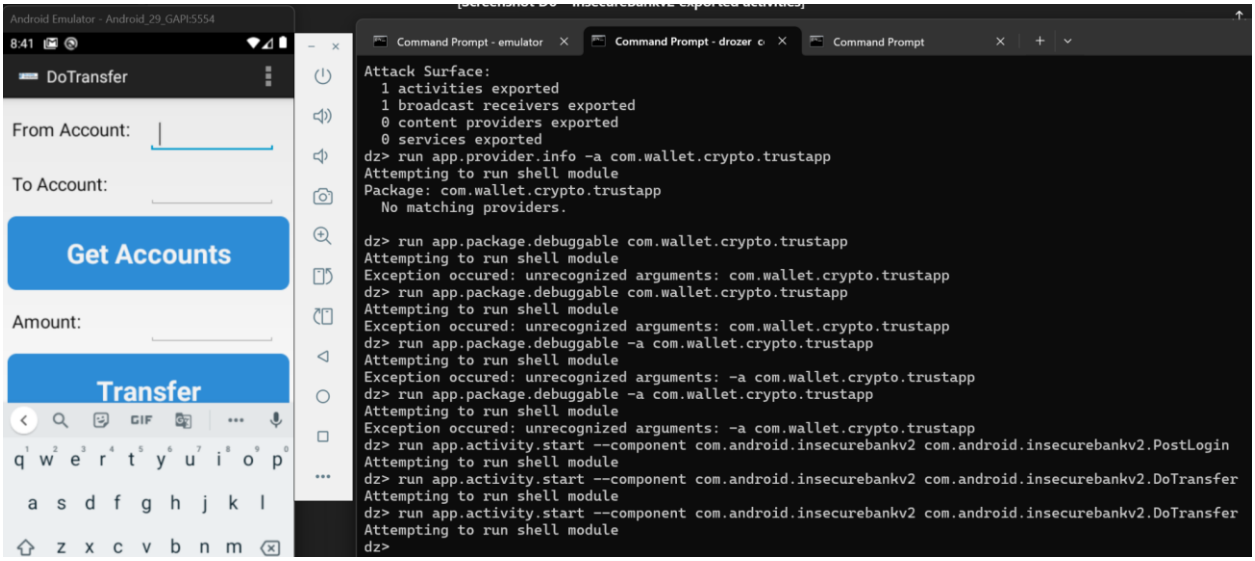


This immediately opened the authenticated home screen without entering any credentials.

I then used similar commands to open other sensitive screens:

run app.activity.start --component com.android.insecurebankv2 com.android.insecurebankv2.DoTransfer

run app.activity.start --component com.android.insecurebankv2 com.android.insecurebankv2.ViewStatement



These activities provide access to banking operations and account information. The app assumes they are only ever reached through the login flow, but because they are exported and unprotected, Drozer (and any other app) can reach them directly.

This is a classic **exported activity vulnerability**:

- Internal screens are exposed with no permission checks.
- Authentication is enforced only in the UI flow, not at the component level.

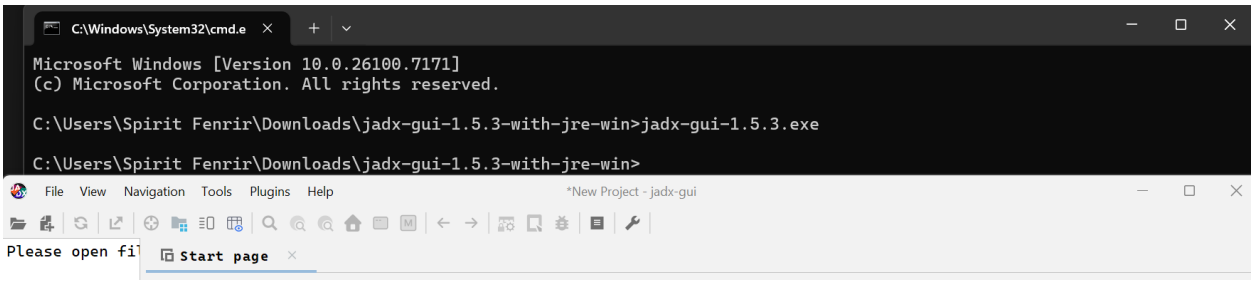
This gives a powerful demonstration of how Drozer can move from **reconnaissance (Spotify, Trust Wallet)** to **actual exploitation (InsecureBankv2)**.

3.10 JADX

JADX is a decompiler for Android APKs that converts Dalvik bytecode back into human-readable Java/Kotlin-like source and reconstructs resources such as AndroidManifest.xml. While Drozer shows what a malicious app can do at runtime, JADX provides a **static view** of the app:

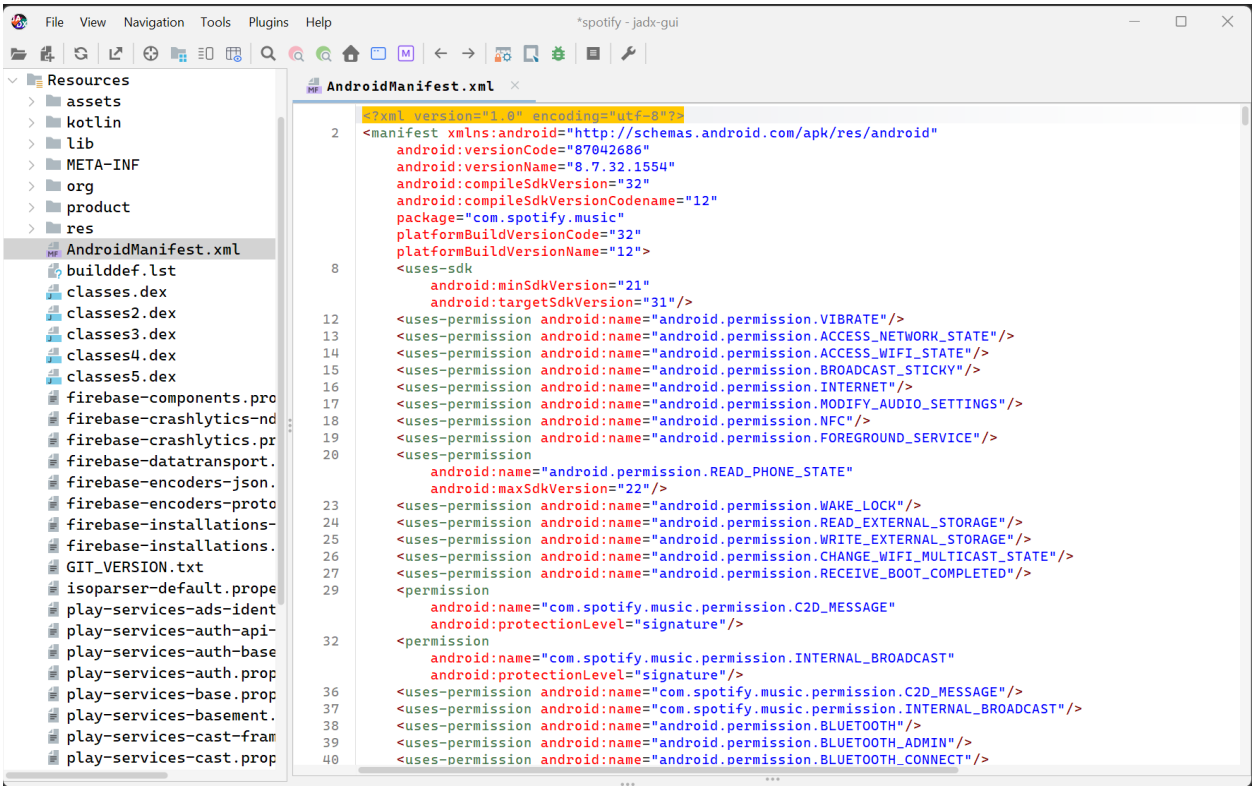
- Manifest (permissions, components, exported flags).
- Package structure and class organization.
- Obfuscation levels.
- Hardcoded URLs, API endpoints, and log messages.

For this project, I used JADX to examine both Spotify and Trust Wallet statically and compare their design choices and security posture.



3.11 Spotify Analysis using JADX

After opening the Spotify APK in JADX, I navigated to AndroidManifest.xml.

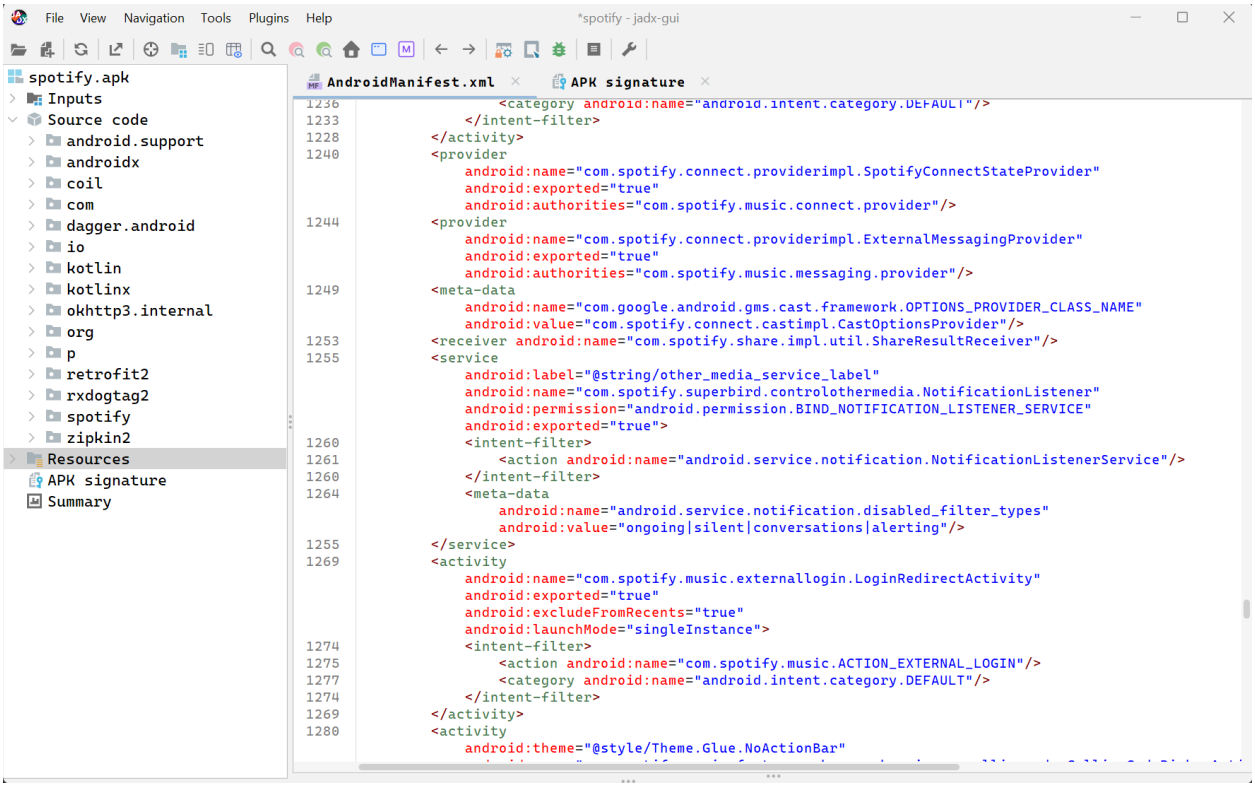


Manifest & Permissions

The manifest showed a large number of permissions typical of a feature-rich multimedia application, for example:

- Network access (INTERNET, ACCESS_NETWORK_STATE)
- Background behavior
- Media and notification-related permissions

Spotify also defined several activities, services, receivers, and providers. Importantly, the **providers for messaging and connect** appeared here as well, including their authorities. This directly matches what Drozer discovered dynamically.



This cross-validation between Drozer and JADX:

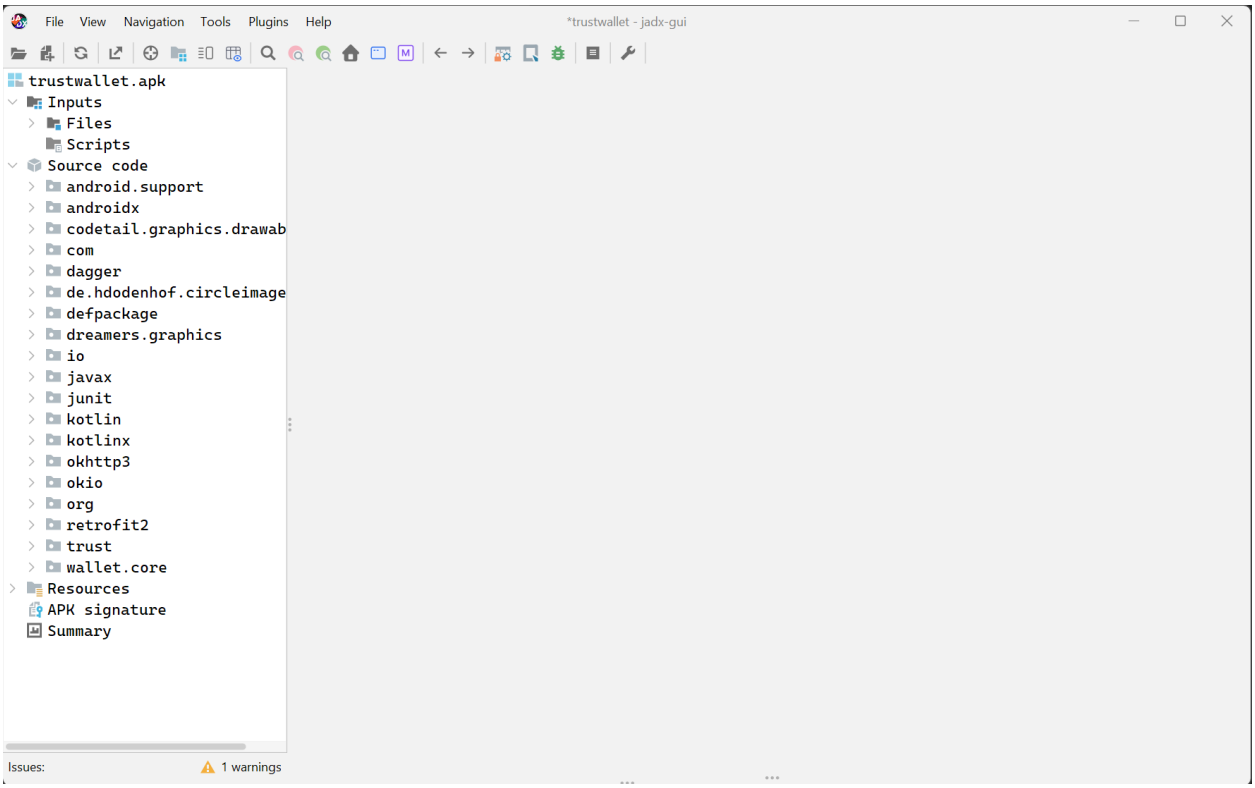
- Confirms that Drozer's provider recon is accurate.
- Shows how static and dynamic analysis complement each other.

Code Structure

Looking at the source tree in JADX, Spotify's codebase was large and split across many packages. Some obfuscation is used (short names and internal packages), but overall, class naming is still somewhat descriptive compared to a fully obfuscated app like Trust Wallet. This reflects a balance between performance, maintainability, and protection.

3.12 Trust Wallet Analysis using JADX

Trust Wallet, being a crypto wallet, places much stronger emphasis on obfuscation and component hardening.

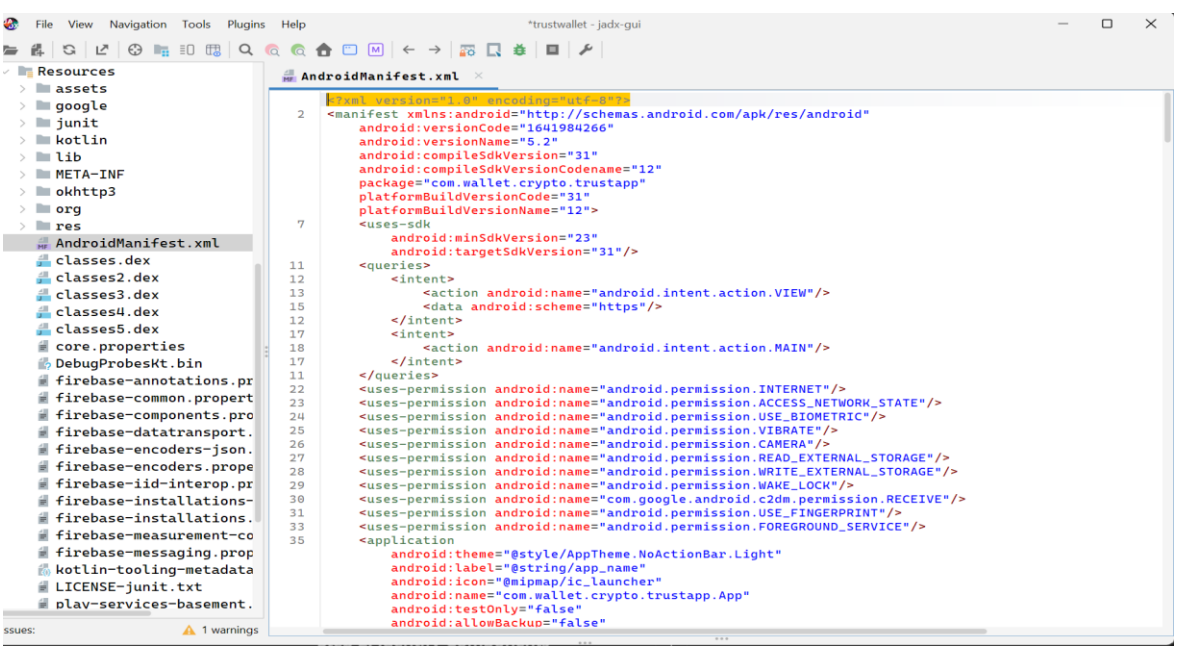


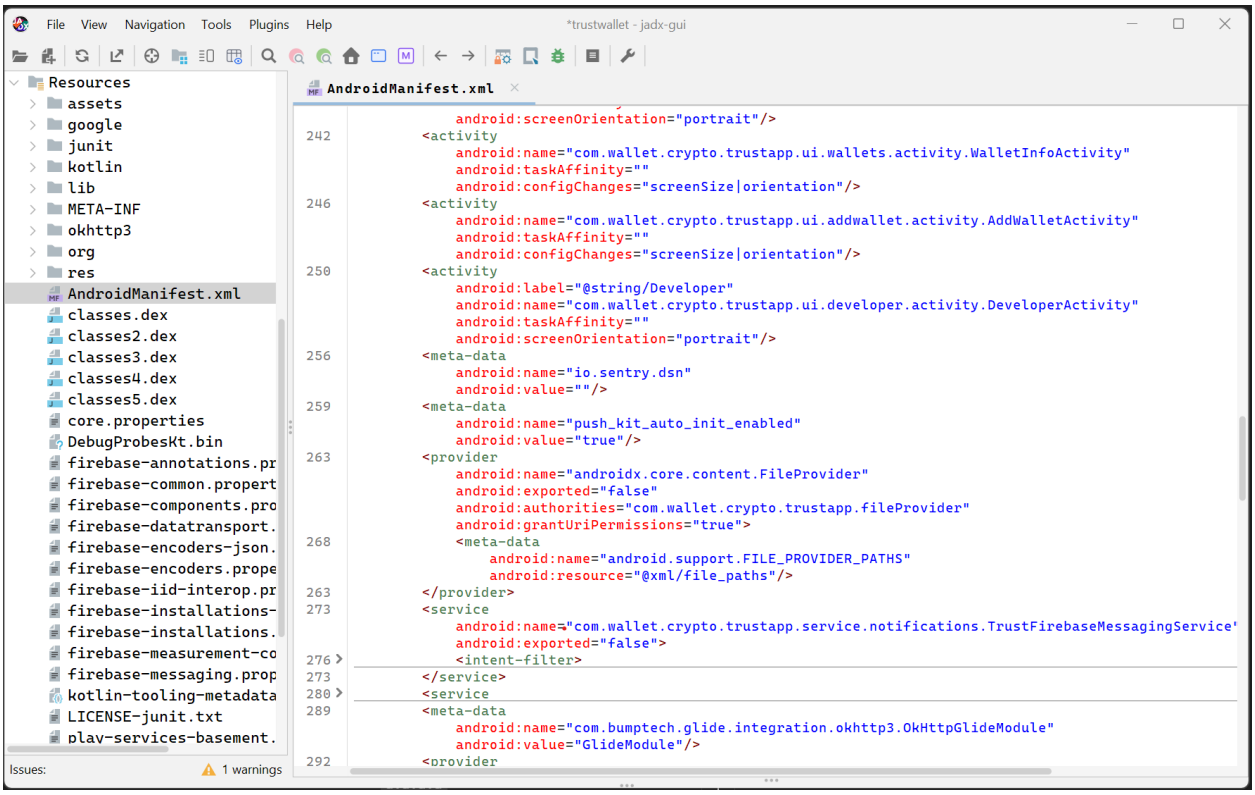
Manifest & Permissions

The AndroidManifest.xml for Trust Wallet showed:

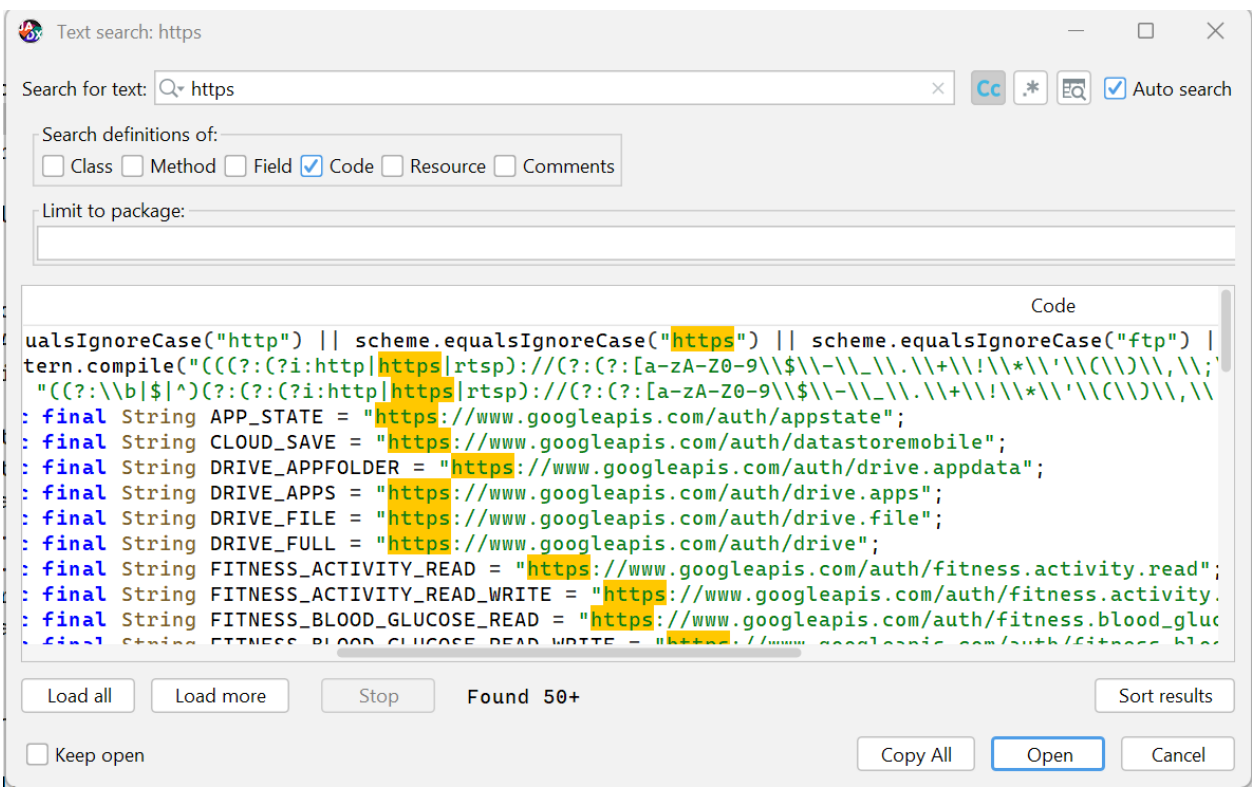
- Standard permissions such as INTERNET and ACCESS_NETWORK_STATE.
- Additional capabilities consistent with managing multiple blockchain networks and notifications.

However, Trust Wallet appeared to expose **very few exported components**. Many internal Activities and Services were defined without being made accessible externally. This aligns with the Drozer observation that Trust Wallet has a minimal attack surface.





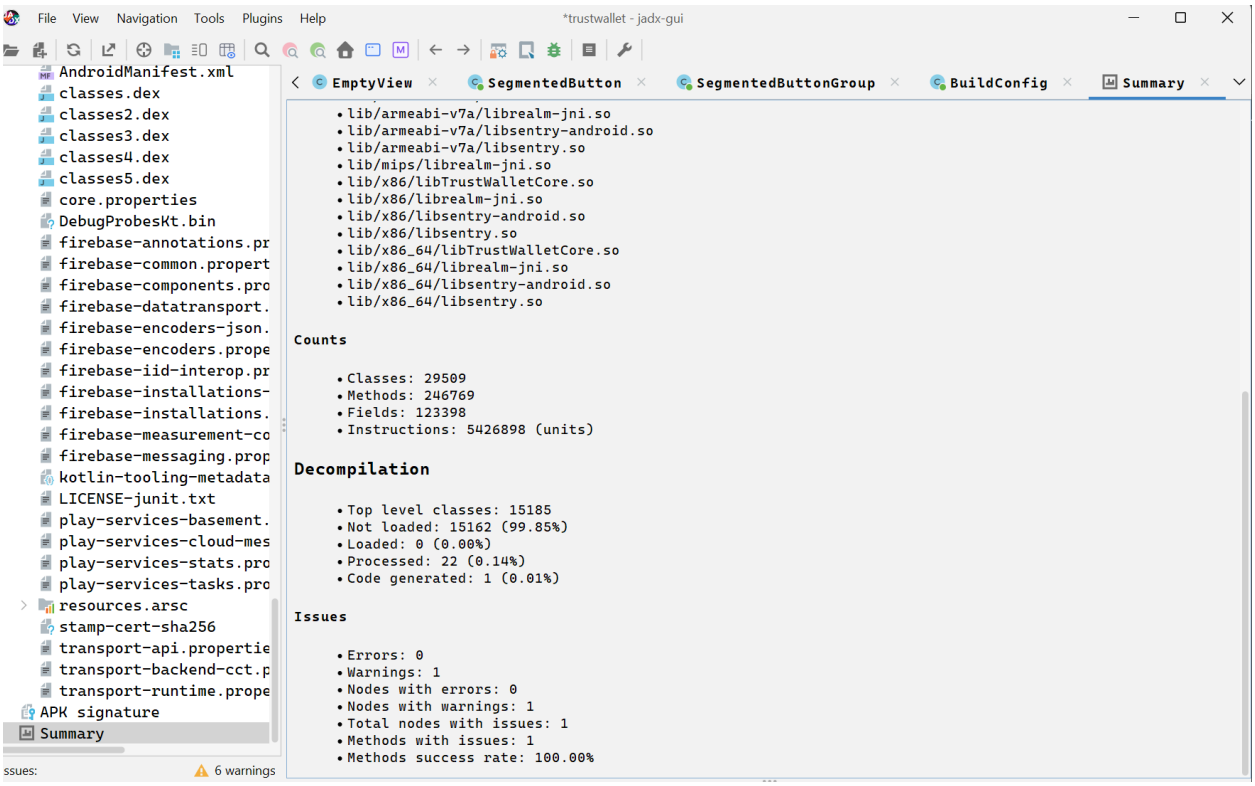
Using jadx's search functionality for strings such as "https", "rpc", and "wallet", I was able to locate some network endpoints and Web3-related strings.



While the exact implementation remains hard to understand due to obfuscation, these strings confirm that Trust Wallet communicates with remote APIs and blockchain infrastructure as expected but without exposing obvious secrets or weak points in plain text.

Overall, JADX reveals that Trust Wallet:

- Intentionally hides its internal logic through obfuscation.
- Keeps exported components to a minimum.
- Does not store sensitive keys or secrets in easily readable form.



Despite containing many classes, the decompilation shows that only a tiny fraction is reconstructed at the AST level, reflecting **optimized or partially obfuscated builds**, requiring analysts to rely more on manifest and string-level recon rather than full code reconstruction.

The app decompiles cleanly with zero runtime errors in the static analyzer, indicating a structurally stable build, while revealing minimal warnings, which influences how static hooks appear to reconnaissance tools like Drozer.

3.13 Integrated Component Prototype

Purpose

To demonstrate team collaboration and avoid fragmented tooling, we created a lightweight automation wrapper that centralizes data acquisition and module planning before moving to dynamic or static analysis.

What the script does

- Accepts a valid device/emulator ID and list of Android package names
- Uses ADB only for acquisition to pull APK paths and exported app data when sandbox permissions allow
- Generates organized forensic output directories for each target app
- Produces a Drozer command cheat-sheet per package, enabling fast internal recon and exploitation demos interactively from the dz> console
- Prepares folders for later JADX decompilation, which are populated manually via a single jadx -d <out> <apk> command per app

Value of integration

This wrapper supports cross-tool examination of the same apps by:

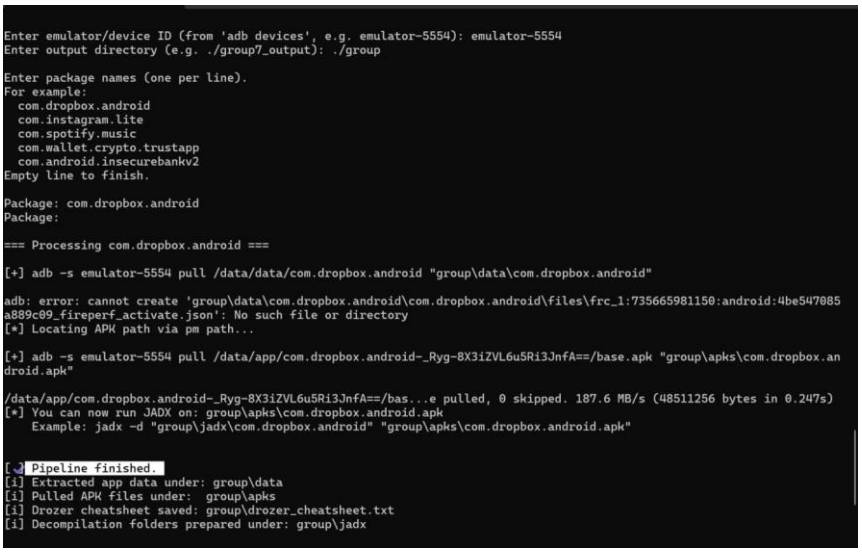
- Providing a reproducible evidence trail
- Reducing command errors and manual repetition
- Allowing each student to inspect runtime IPC surface (Drozer) and static architecture (JADX, ALEAPP, MobSF) from a unified app list
- Real forensic insight from tooling

During execution we observed Android sandbox restrictions in action:

- Production apps such as Spotify and Trust Wallet restrict /data/data extraction unless the app or build image allows root or run-as privileges
- The vulnerable training app InsecureBankv2 can populate acquired data and APK paths reliably
- These permission denials validate Android's component isolation and form a legitimate forensic observation on tool limits

Conclusion for this component

"The integrated pipeline does not bypass Android security itself; it automates acquisition and organizes app components for reliable IPC-level analysis (Drozer) and APK static inspection (JADX and MOBSF).



```
Enter emulator/device ID (from 'adb devices', e.g. emulator-5554): emulator-5554
Enter output directory (e.g. ./group7_output): ./group

Enter package names (one per line).
For example:
  com.dropbox.android
  com.instagram.lite
  com.spotify.music
  com.wallet.crypto.trustapp
  com.android.insecurebankv2
Empty line to finish.

Package: com.dropbox.android
Package:

=== Processing com.dropbox.android ===

[+] adb -s emulator-5554 pull /data/data/com.dropbox.android "group\data\com.dropbox.android"

adb: error: cannot create 'group\data\com.dropbox.android\com.dropbox.android\files\frc_1:735665981150:android:4be547085
a889c09_fireperf_activate.json': No such file or directory
[*] Locating APK path via pm path...

[+] adb -s emulator-5554 pull /data/app/com.dropbox.android-_Ryg-8X3iZVL6u5Ri3JnfA==/base.apk "group\apks\com.dropbox.an
droid.apk"

/data/app/com.dropbox.android-_Ryg-8X3iZVL6u5Ri3JnfA==/bas...e pulled, 0 skipped. 187.6 MB/s (48511256 bytes in 0.247s)
[*] You can now run JADX on: group\apks\com.dropbox.android.apk
    Example: jadx -d "group\jadx\com.dropbox.android" "group\apks\com.dropbox.android.apk"


[✓] Pipeline finished.
[i] Extracted app data under: group\data
[i] Pulled APK files under:  group\apks
[i] Drozer cheatsheet saved: group\drozer_cheatsheet.txt
[i] Decompilation folders prepared under: group\jadx
```

4. Results and Insights

4.1 ALEAPP

The ALEAPP analysis produced a comprehensive picture of the device's activity, user behavior, and installed apps. Several artifacts revealed clear user identity links, including the Gmail address **csis4440user@gmail.com**, Google Calendar setup, and Maps "Home" and "Work" labels with exact coordinates tied to Vancouver and Burnaby. Communication logs were also significant two call logs, two SMS messages, and three saved contacts were recovered, showing recent interactions and confirming that the device had active telephony usage. System-level identifiers such as the Android ID, IMSI, ICCID, device Bluetooth MAC, and hotspot credentials provided additional technical insight into the device configuration and network usage patterns.

The multimedia and storage metadata offered strong contextual evidence of user activity, including downloaded images, screenshots, Instagram-generated media, and APK packages stored in the device's Downloads folder. Digital Wellbeing logs produced over 600 entries, giving a detailed timeline of app launches, notifications, and device events such as shutdowns and startups useful for reconstructing chronological behavior. Installed app data confirmed the presence of key applications used in this investigation (Dropbox, Instagram Lite, Pinterest) along with their version codes and SHA-256 hashes. Collectively, these findings show that ALEAPP can reconstruct user identity, movement, communication, media interactions, and application usage from a single full-device extraction, demonstrating strong forensic value.

4.2 MobSF (Static Analysis)

The MobSF static analysis of the Pinterest APK clearly demonstrated the tool's ability to break down an application's internal structure, security posture, and behavioral components without needing to run the app. One of the most significant outcomes was MobSF's extraction of the **AndroidManifest.xml**, which revealed over 30 declared activities, 17 background services, 19 broadcast receivers, 3 content providers, and multiple embedded libraries. This highlighted how MobSF helps investigators understand an app's internal architecture and identify components that may interact with user data or external systems.

MobSF also provided a detailed breakdown of **requested permissions**, automatically classifying them by severity. Dangerous permissions such as *camera access, fine location, account listing, contacts reading*, and multiple media access permissions were flagged, demonstrating how MobSF helps assess potential privacy and forensic relevance. The tool also identified **network security configurations**, warning that the base configuration allows clear-text traffic while confirming TLS enforcement for Pinterest's major domains.

Another strong insight was MobSF's ability to detect **hardcoded secrets**, including API keys, tokens, internal URLs, error messages, and configuration flags. In the Pinterest APK, the tool identified over 900 strings that may be sensitive, showing how MobSF can uncover embedded resources that might pose risks or assist in further forensic investigation. Additionally, MobSF extracted the application's **signing certificate**,

confirming the authenticity of the APK, verifying v2/v3 signature schemes, and revealing metadata such as the certificate issuer, validity window, and hashing algorithms used.

Overall, the MobSF results demonstrate how the tool provides a powerful, automated, and comprehensive static analysis pipeline. It reveals internal components, permissions, network rules, embedded secrets, certificates, and other critical artifacts making it extremely valuable for both mobile security assessment and mobile forensic investigations.

4.3 Dropbox

The Dropbox analysis produced rich and highly structured evidence, as the application stores detailed metadata in multiple SQLite databases. The *dropbox.db* database revealed full lists of synced files, including file names, canonical paths, MIME types, parent folders, timestamps, and thumbnail existence. This allowed reconstruction of the exact folder structure used by the account (e.g., *School*, *Work*, *Medical*, *Photos*, *Projects*). The photos, thumbnail_info, and recents_entries tables confirmed recent activity, such as files viewed or synced, images accessed, and timestamps showing when specific photos or documents were interacted with. The starred_info table showed items manually marked as important, while the comments table preserved user-generated comments linked to particular files. Combined, these datasets provide a comprehensive view of user behaviour uploaded materials, file organization habits, active project timelines, and which resources the user prioritized or interacted with most.

Overall, Dropbox produced some of the most valuable forensic artifacts in the entire extraction. The presence of thumbnails, recent activity logs, and detailed paths enables accurate reconstruction of user actions, including document access patterns, photo uploads, and file management behaviour. These insights reflect clear usage of Dropbox for academic, work, and personal file storage, including cybersecurity material, receipts, PDFs, and school documents. The quality and quantity of Dropbox metadata make it ideal for timeline reconstruction and behavioural profiling.

4.4 Instagram Lite

Instagram Lite generated significantly fewer structured artefacts, consistent with its design as a lightweight app with minimal on-device storage. A manual review of the extracted */data/data/com.instagram.lite* folder revealed mostly configuration files, caches, and webview structures, with no accessible user messages, profiles, or feed interactions stored in SQLite form. However, ALEAPP uncovered **1,402 cached image fragments** stored under */cache/image_cache/v2.ols100.1*, representing thumbnails and media displayed to the user. These include pieces of feed images, profiles, and other visual elements processed by the app, providing the primary source of user-related artefacts.

ALEAPP's **Timeline report** further confirmed Instagram Lite activity by surfacing cookie entries (e.g., authorization tokens, session IDs, device IDs) and detailed timestamps associated with cache item creation. These timeline entries validated when the user viewed specific content, even though the full images were not stored. Collectively, the cached images and timeline cookies provide partial but meaningful insight into app usage, demonstrating that while Instagram Lite limits persistent user data, **its cache still reveals viewing activity and session-related metadata** useful for reconstructing interactions.

4.5 Strava

Before the authentication failures and crashes began, a partial extraction of Strava's internal SQLite databases revealed meaningful profile-level forensic artifacts. While activity data never populated, several tables including logged_in_athlete, athletes, and related metadata structures contained rich JSON objects reflecting the user's demographic and profile configuration.

The most notable artifact was found in **logged_in_athlete**, which stored an extensive JSON profile of the authenticated user. This included:

- Full name: **CSIS440 User**
- Username: **csis440_user**
- Country & city: **Vancouver, Canada**
- DOB: **1997-11-16**
- Gender: **M (man)**
- Weight: **63.0 kg**
- Email: **csis4440user@gmail.com**
- Measurement preference: **meters**
- Privacy settings: visibility = *everyone*
- Max heart rate: **193**
- First login timestamp (created_at): **2025-10-30T04:42:08Z**

This demonstrates that Strava stores sensitive user metadata including health-related information in plaintext JSON within the app's local databases, making them highly relevant from a forensic perspective.

Other tables such as segments, weekly_stats, clubs, gear, and activities were created but empty, confirming that the app skeleton was functional but never synchronized actual workout activities into the emulator. Even the *"all_time_activity_count": 0* value reinforces that no activities had been synced.

Additionally, multiple entries under **subscription_detail**, **progress_goals**, **map_treatments**, and **notifications** showed that Strava pre-populates layout and configuration data regardless of actual activity use. However, these were not connected to live activity events.

In summary, despite the emulator never receiving activities, the database extraction proved that:

1. The login initially succeeded.
2. Profile-level forensic evidence **was** stored and accessible.
3. Strava would have produced rich artifact sets if synchronization had not failed later.

4.6 Drozer, JADX, Spotify, TrustWallet & InsecureBankv2

- **Spotify:** IPC providers detected, schema tables visible but hardened against credential leakage
- **Trust Wallet:** Minimal attack surface, rejects debugging attach, strong component isolation

- **InsecureBankv2:** Fully exploitable exported Activities allowing authentication bypass and direct access to sensitive banking UI

How was the exploration useful?

- Drozer demonstrated how Android apps expose or restrict Activities, Providers, Broadcast Receivers, and Debug surfaces at runtime from an attacker context.
- JADX allowed cross-validation of provider authorities enumerated in Drozer, confirming static vs dynamic visibility.
- Spotify gave a real-world reconnaissance case, Trust Wallet gave a crypto-hardened contrast, and InsecureBankv2 showed live exploitation impact.

Limits of research / exploration

- Spotify DB is modular and fragmented, not monolithic, making targeted table-analysis required
- Trust Wallet shows restricted IPC pathways, so recon is possible but exploitation is prevented
- Drozer modules must be used with -a flags for app context
- Certain exploit modules don't return output on hardened packages
- Unknown modules may appear during exploration, showing limits of module availability
- Static analysis cannot recreate 100% of bytecode into readable AST, especially when apps include native .so libraries

5. Conclusions

5.1 Tools & Applications - Umesh

The forensic investigation successfully demonstrated how different tools can extract and interpret data from Android applications, even when the level of recoverable evidence varies across apps. Dropbox provided rich and well-structured artifacts, including thumbnails, cached folder structures, file metadata, recently accessed media, and database entries showing that cloud-storage apps retain significant local traces that are highly valuable in digital forensics. In contrast, Instagram Lite stored minimal user-level data, but the use of ALEAPP revealed over 1,400 cached image fragments and a detailed timeline of cookie activity and media access events, confirming that even lightweight social media apps can leave behind meaningful artifacts.

The use of multiple tools was essential to obtain a complete picture. ADB extraction enabled full access to app directories, DB Browser supported manual database reviews, and ALEAPP automated the parsing of hundreds of files across images, logs, caches, and webview artifacts. MobSF further contributed by providing a deep static analysis of an APK, highlighting permissions, component structure, hardcoded secrets, and network security behavior. Together, these tools showed how a layered approach manual inspection combined with automated parsers and static application analysis provides the most reliable results. Overall, the combination of **Dropbox, Instagram Lite, and MobSF**

demonstrated the strengths and limitations of mobile forensic methods and emphasized the importance of using the right mix of extraction and analysis tools for accurate, comprehensive investigations.

5.2 Tools & Applications - Sahan

Future Work

- Spotify Connect and messaging provider inputs using dynamic agent modules
- Explore native .so code segments found by static decompile
- Extend analysis to WalletConnect deep link intent handling for Trust Wallet
- Build a detection module set for crypto RPC communication visualization

Other scenarios where these insights apply

- Banking apps with exported internal Activities
- Social apps exposing device/session providers
- Wallet applications limiting IPC but potentially exposing backend strings
- Incident analysis for phishing, exported intent abuse, or sandbox privilege escalation testing

Broad conclusion

Drozer excels at runtime IPC attack-surface mapping and malicious-activity simulation, while JADX complements with static manifest and communication-string exposure. Combining the same apps across both tools enables powerful comparisons and real-world capability demonstrations without requiring filesystem extraction.

6. AI Use Section

Table of AI Tools and Specific Use:

AI Tool Name	Version, Account Type	Specific feature for which AI tool was used
ChatGPT	GPT-5 (Plus)	Assisted in creating GPX sample data, explaining emulator errors, interpreting logcat output, and guiding configuration steps for ADB, MobSF, and ALEAPP. Helped troubleshoot extraction issues, identify relevant command sequences, and clarify Android file-system artifacts. Provided

		explanations of forensic concepts (e.g., caches, cookies, WebView storage), recommended analysis paths, refined report wording, and summarized findings. Also used to understand MobSF static analysis output, permissions, manifest structure, and APK components.

Although AI tools were used to guide troubleshooting, generate documentation, and simplify explanations, all final decisions, implementations, and forensic workflows were performed manually by me. This includes:

- Manual environment setup - installing ADB, pulling /data/data directories, configuring MobSF with Docker.
- Performing all extraction and acquisition steps using ADB and filesystem access.
- Hands-on use of forensic tools - ALEAPP, DB Browser for SQLite, MobSF to view, verify, and interpret each artifact.
- Independent analytical decisions about what evidence was relevant, how data should be interpreted, and how conclusions were formed.
- Writing, editing, validating, and verifying the entire report, ensuring originality and technical accuracy.
- Testing cycles and validation, including re-running extractions, confirming timestamps, checking cache directories, and ensuring forensic repeatability.

AI provided suggestions and explanations, but I performed all technical actions, verifications, testing cycles, file manipulations, and forensic validation myself, ensuring the analysis remained academically original and hands-on.

7. Work Log Table

Date	Student	Work Summary	Hours
15th Sep 2025	Both	Group formed and agreed on individual roles.	0.5
6th Oct 2025	Umesh	Researched tools and apps suitable for the project.	1.0
6th Oct 2025	Sahan	Conducted research on tools and applications for analysis.	1.0
7th Oct 2025	Umesh	Drafted an email to professor to finalize tools and applications.	0.5
9th Oct 2025	Both	Met with professor to discuss project scope and confirm selected tools & apps.	0.5

21st Oct 2025	Umesh	Researched and set up ALEAPP and MobSF; drafted extraction workflow for Dropbox & Strava.	1.0
22nd Oct 2025	Sahan	Researched and set up Drozer and JADX; drafted assessment plan for Trust Wallet & Spotify.	1.0
28th Oct 2025	Umesh	Installed Android SDK command-line tools and configured AVD, ADB, SDKManager, and APKAnalyzer paths. Created and launched emulator. Installed Dropbox and Strava APKs for testing.	2
29th Oct 2025 (Night)	Umesh	Created an email account, separate Strava and Dropbox test accounts and linked them to emulator. Generated a mock GPX route around Vancouver and imported it to simulate an activity. Activity did not appear on Strava; began troubleshooting and analyzing logs.	2.0
30th Oct 2025 (Morning)	Umesh	Investigated Strava database structure using DB Browser for SQLite. Injected mock JSON activity entry into strava-database. Encountered app crash due to missing native library (libFatmapSdk.so).	2.5
1st Nov 2025 (Morning)	Sahan	Setup drover and made a successful connection to the emulator. encountered error connecting it to the emulator and getting the right ip address	3
5th Nov 2025	Umesh	Uploaded prepared sample files to Dropbox to simulate user data and activity. Verified folder structure for forensic testing.	1.0
5th Nov 2025	Umesh	Installed and configured both ALEAPP and MobSF tools in the project environment. Verified successful setup and readiness for analysis.	1.5
6th Nov 2025	Sahan	Tested Drozer connectivity to an Android emulator, learned how to start the Drozer agent, and explored basic commands (run app.package.list, run app.package.info).	1.0
7th Nov 2025	Umesh	Attempted to create sample app data for Strava using emulator.	1
8th Nov 2025	Sahan	Installed JADX-GUI and JADX-CLI, explored interface layout, and learned preferences for decompilation accuracy.	1.5
8th Nov 2025	Umesh	Created sample activities for Strava account using a separate device and made it visible in the emulator.	1
9th Nov 2025	Umesh	Pulled data/data directories of both the apps and ran ALEAPP on those folders.	1.5
9th Nov 2025	Sahan	Practiced decompiling a sample APK using JADX, inspected file tree, learned how to navigate Java source, resources, and smali.	2.0
9th Nov 2025	Sahan	Researched Drozer modules for attack surface mapping (scanner.activity, scanner.provider, scanner.broadcast).	1.5

9th Nov 2025	Umesh	Analyzed permission errors for .mapbox & .fit files in Strava extraction. Reran with elevated privileges.	1
9th Nov 2025	Umesh	Troubleshoot empty ALEAPP reports and manually inspected WAL/SQLite files for forensic artifacts.	2
10th Nov 2025	Sahan	read articles on common Java decompilation issues.	1.0
11th Nov 2025	Umesh	Attempted MobSF local server run via python and failed, then began docker setup for MobSF container	1.5
12th Nov 2025	Umesh	Attempted dynamic analyzer configuration in MobSF. Encountered Android Runtime detection errors, emulator connection failures, and ADB remount issues. Verified that emulator was detected by MobSF but system partition was not writable.	1
12th Nov 2025	Umesh	Investigated MobSF dynamic analysis requirement for API level 30-based emulators and created a fresh Android Studio emulator running Android 11 (API 30) to meet the minimum supported version by MobSF.	0.5
12th Nov 2025	Umesh	Attempted to run dynamic analysis again with API 30 emulator, but MobSF reported system partition as read-only.	1
12th Nov 2025	Umesh	Switched to static analysis workflow. Extracted APK files directly from emulator using ADB. Identified package locations, tested multiple extraction commands, and prepared APKs for Strava & Dropbox for MobSF static analysis.	1
13th Nov 2025	Sahan	Practiced identifying exported components using Drozer; tested simple exploitation of exported activities using Drozer commands.	1.0
15th Nov 2025	Sahan	Prepared notes for progress report.	2.0
16th Nov 2025	Sahan	Continued dynamic IPC reconnaissance using Drozer; tested additional modules including app.broadcast.info, scanner.misc.readablefiles, and learned differences in syntax for package-level modules.	1.5
17th Nov 2025	Sahan	Used Drozer to enumerate providers for Spotify and Trust Wallet; verified provider accessibility and tested queries on both connect and messaging authorities.	2.5
17th Nov 2025	Umesh	Began analyzing Dropbox data extracted earlier via ADB. Opened the internal SQLite databases using DB Browser and reviewed folder metadata, sync records, and cached file paths. Added additional sample files to Dropbox to generate richer forensic artifacts.	2.0

17th Nov 2025	Umesh	Investigated Strava login failure. Cleared app data, reinstalled APK, modified network settings. OTP verification consistently failed.	1.5
18th Nov 2025	Sahan	Studied Intent handling capabilities using app.activity.forintent on exported components from vulnerable app; learned how exported Activities respond to unprivileged IPC calls.	1.5
18th Nov 2025	Umesh	Attempted Strava login on both physical phone and emulator. Physical login succeeded briefly, but emulator repeatedly rejected OTP. Captured and reviewed logcat errors.	1.0
19th Nov 2025	Sahan	Ran JDWP exposure scan (exploit.jdwp.check) to verify debugging attachment posture for Trust Wallet; documented behavior where detection is possible without exploit path.	2.1
19th Nov 2025	Umesh	Re-opened previously extracted Strava databases. Verified presence of personal profile data (height, weight, DOB, city). Determined further data acquisition impossible without account access.	1.5
20th Nov 2025	Sahan	Installed InsecureBankv2 on emulator and mapped full attack surface using Drozer; drafted exploitation flow points to showcase authentication bypass.	1.7
20th Nov 2025	Umesh	Performed ALEAPP parsing on the entire emulator /data/data directory and analyzed extracted files. Identified SQLite fragments, WAL files, and partial Strava traces. Confirmed most Strava artifacts were incomplete due to failed login.	1.0
21th Nov 2025	Sahan	Performed the main exploit demo using app.activity.start to launch PostLogin, bypassing authentication; validated access to sensitive banking UI without login.	1.8
21st Nov 2025	Umesh	Created a TAR archive of the emulator's /data/data folder for consistency. Re-ran ALEAPP on TAR output to ensure full parsing. Final attempt to inject GPX/JSON mock activities into Strava DB; process failed due to app-side verification checks. Officially documented the Strava abandonment decision.	1.0
22th Nov 2025	Sahan	Opened Spotify APK in JADX-GUI; inspected AndroidManifest.xml for activity, provider, broadcast, and permission declarations for static cross-validation with Drozer results.	2.5
22nd Nov 2025	Umesh	Conducted static analysis of the Pinterest APK using MobSF. Reviewed permissions, domain configurations, trackers, signature details, and hardcoded secrets. Summarized MobSF findings for report.	1.5

23th Nov 2025	Sahan	Opened Trust Wallet APK in JADX; navigated sources and confirmed this build is not aggressively obfuscated; used global string search to identify network communication endpoints.	2.5
24th Nov 2025	Sahan	Documented APK complexity from JADX Summary view; inspected native library architectures (ABI folders + .so variants).	2.0
24th Nov 2025	Umesh	Installed Instagram Lite APK on emulator. Created a fresh test account and performed typical user actions (viewing posts, navigating feed) to generate forensic data. Verified app stability compared to original Instagram app.	1.0
24th Nov 2025 (Evening)	Umesh	Extracted /data/data/com.instagram.lite using ADB. Prepared folders for ALEAPP ingestion.	1.0
25th Nov 2025	Sahan	Compared manifest export posture between Spotify and Trust Wallet in JADX with dynamic results from Drozer to form report insights.	1.0
25th Nov 2025	Umesh	Ran ALEAPP on the extracted Instagram Lite folder. Recovered 1402+ cached images and metadata. Exported and validated HTML reports.	1.5
25th Nov 2025 (Night)	Umesh	Inspected Instagram Lite's internal SQLite databases using DB Browser. Confirmed minimal local storage (mostly cache paths and small configuration tables). Documented findings.	1.0
26th Nov 2025	Sahan	Practiced extracting interesting presentation points focusing on tool capability, sandbox posture, exported component abuse, and real-world recon limits.	1.0
27th Nov 2025	Umesh	Drafted results & insights sections, prepared documentation, and organized screenshots for the report.	2.0
27th Nov 2025	Sahan	Organized findings into report sections, captured screenshots and verified visual placements according to template.	2.2
28th Nov 2025	Both	Created Archival Video and finalized the documentation.	2.5

8. Appendix

Prompts - Umesh

“I need to create a mock gps route around Vancouver in .gpx format”

“Error: adb connect host.docker.internal:5555 explain the cause and fix for MobSF dynamic analysis.”

“Explain how to remount /system as writable in an Android emulator.”

“Why does MobSF say ‘VM’s /system is not writable’ and how can I fix this?”

“How do I run a static analysis on an APK using MobSF if dynamic analysis is not working?”

“ALEAPP produced empty reports for Strava and Dropbox. What files do I need to include for successful artifact extraction?”

“MobSF dynamic analysis shows Android Runtime not found, what causes this?”

“What are the best apps to test MobSF’s static and dynamic capabilities?”

“How to fix ‘TypeError: NoneType does not support item assignment’ during MobSF startup?”

“How do I use ALEAPP to parse a full device dump instead of a single app folder?”

Prompts - Sahan

“Explain what IPC means in Android and how Drozer leverages it without requiring root.”

“List the important Drozer modules for analyzing app attack surfaces and exported components.”

“Why do some Drozer commands not bring the launched activity instantly to the emulator foreground?”

“Give me a 3-minute live demo flow to confidently present Drozer findings.”

“Use Drozer to map Spotify’s exported activity, broadcast, and provider surface and describe what each part means for reconnaissance.”

“Suggest one interesting non-crashing finding on Spotify that shows real-world component design.”

“Using Drozer, assess Trust Wallet’s attack surface and debug posture; explain the difference in security expectations for wallet apps.”

“How to find backend URLs or RPC endpoints in JADX without pulling databases using adb?”

“Interpret the security risk when app.activity.start can directly launch PostLogin on InsecureBankv2 without permissions.”

9. References

9.1 Tools

Brignoni, A. (2020). ALEAPP – Android Logs, Events, and Protobuf Parser [Computer software]. GitHub. https://github.com/abrignoni/ALEAPP

Mobile Security Framework (MobSF). (2024). Mobile Security Framework (MobSF) [Computer software]. GitHub. https://github.com/MobSF/Mobile-Security-Framework-MobSF

FSecure Labs. (2015). Drozer [Computer software]. GitHub. https://github.com/FSecureLABS/drozer

Skylot. (2024). JADX – Dex to Java decompiler [Computer software]. GitHub. https://github.com/skylot/jadx

Androguard Project. (2024). Androguard – Reverse engineering and malware analysis of Android applications [Computer software]. GitHub. https://github.com/androguard/androguard

504ensics Labs. (2014). LiME – Linux Memory Extractor [Computer software]. GitHub. https://github.com/504ensicsLabs/LiME

9.2 Apps

Dropbox. (2024). Dropbox (Android app) [Mobile application]. Google Play Store. https://play.google.com/store/apps/details?id=com.dropbox.android

Strava. (2024). Strava (Android app) [Mobile application]. Google Play Store. https://play.google.com/store/apps/details?id=com.strava

Trust Wallet. (2024). Trust Wallet (Android app) [Mobile application]. Google Play Store. https://play.google.com/store/apps/details?id=com.wallet.crypto.trustapp

Spotify. (2024). Spotify (Android app) [Mobile application]. Google Play Store. https://play.google.com/store/apps/details?id=com.spotify.music

Telegram FZ-LLC. (2024). Telegram (Android app) [Mobile application]. Google Play Store. https://play.google.com/store/apps/details?id=org.telegram.messenger

Instagram Lite. Google Play Store https://play.google.com/store/apps/details?id=com.instagram.lite&hl=en_CA

END OF DOCUMENT