Douglas College

Module Code:

CSIS 4440 – 003

Module Title:

Mobile Cybersecurity

Year & Semester:

2025 Fall

Project Final Report

Instructor: Priya

Document Control	
Group	10
Course	4440-003 Mobile Cybersecurity
Members	Allwyn Mascarenhas Chetan Kaur
Team Lead	Allwyn Mascarenhas
Student ID	Allwyn-300381351 Chetan - 300391004
YouTube Link(Allwyn) **YouTube Link(Chetan)**	https://youtu.be/iJJO9kB53MY?si=6k5i_lK_x2W6wFGz https://www.youtube.com/watch?v=W4lj_cc2p7A
GitHub	https://github.com/allwynmas/F25-4440-S003-G10
Report Section	Following is the report by team members Allwyn Mascarenhas and Chetan Kaur

Table of Contents

1. Overview

This report presents findings, testing procedures, and supporting documentation for the evaluation of an Android APK application's security using the drozer Security framework. It details the setup of the testing environment, including the installation and configuration of both the drozer console and drozer agent on a virtual Android device.

The report highlights the steps taken to install required tools, install the APK for assessment, and initiate the penetration testing process using various drozer modules designed for information gathering, attack surface exploration, and vulnerability identification.

Screenshots and documentation are provided to demonstrate each phase of the assessment and substantiate the findings associated with the tested application.

The final demo is hosted on YouTube: https://youtu.be/iJJO9kB53MY

1.1 Tooling and Applications

1.1.1 Drozer

Drozer enables users to assume the role of an Android application, allowing them to interact with the Android Runtime, discover vulnerabilities in apps and devices, and probe other apps' IPC endpoints and the underlying operating system. It is effective for mimicking how a malicious application could exploit weaknesses, supporting real-world attack simulations without needing custom apps—users can issue commands via the Drozer console.

Key capabilities include:

- Identification of a broad range of vulnerabilities (e.g., insecure storage, privilege escalation, insecure communications).
- Tools for using, sharing, and understanding public Android exploits, as well as automating regression testing through scripting and extension modules.
- Assessment of Android apps and devices in production mode, because it doesn't require USB debugging or other dev features, supporting both emulators and real hardware.
- Fully open source and actively maintained by Reversec.

1.1.1.1 Drozer PC Client/Console

The Drozer Client is the PC-based component that acts as the command-and-control interface for interacting with the testing device. Through the Client, testers issue commands, manage

modules, and receive results. It connects with the Agent on the Android device to carry out penetration tests and security assessments.

Main roles:

- Provides a user-friendly console for managing tests and modules.
- Enables seamless communication with the Agent over the network or USB.
- Downloadable from the official project repository and release pages.

1.1.1.2 Drozer APK Agent

The Drozer Agent is an Android APK installed on the target device or emulator to facilitate security testing. It acts like a lightweight app, capable of interacting directly with the Android environment, app components, and the operating system from the perspective of a standard application.

Main roles:

- Executes commands received from the Drozer Client, running tests and exploits as instructed.
- Simulates an actual app's behaviour, including interacting with IPC mechanisms and exposed interfaces.
- Does not require special debugging or developer settings, allowing use on everyday devices in their standard state.
- Designed for easy deployment and removal, so devices can be quickly returned to normal operation after testing.

1.1.2 AndroGOAT

AndroGoat is a purposely developed open-source vulnerable/insecure application using **Kotlin**. It serves as a learning tool for security professionals and developers to understand, exploit, and defend against vulnerabilities in the Android platform.

As the first vulnerable app developed natively in Kotlin, AndroGoat is the perfect solution for anyone looking to master modern Android Application Security Testing.

2. Demo Environment Setup

The details of the environment used in this demo are as follows:

Component	Version / Details
Host Operating System	Windows 11

Android APK Analysis Tool	Drozer 3.1
Drozer Agent APK (installed in Android emulator)	Drozer Agent v3.1.0
Android Studio (for emulator)	Android Studio 2024.3.1Meerkat
Docker	Docker Desktop v4.52.0
APK Static Analysis/Reverse Engineering	JadX v1.5.3
Target App (for exploitation)	AndroGoat v2.0.1

2.1 Drozer Tool Setup

Install Docker Desktop

- Download & install Docker Desktop for Windows 11
- Ensure **WSL2 backend** is enabled

Pull Drozer Docker Image

- Run:
  docker pull drozerdocker/drozer

Install Android Studio (Meerkat 2024.3.1)

- Install Android Studio
- Install required **Android system image**:
  sdkmanager "system-images;android-36;google_apis;x86_64"

Create and Launch Emulator

- Create AVD:
  avdmanager create avd -k "system-images;android-36;google_apis;x86_64" -n AndroidDrozer
- List devices (optional):
  avdmanager list devices
- Start emulator:
  emulator -avd AndroidDrozer

Install Drozer Agent (APK)

- Install APK on emulator from adb:
  adb install drozer-agent.apk
- Open the app → **Enable** the agent click the ON button

ff

Set Up ADB Port Forwarding

- Forward ports:
  adb forward tcp:31415 tcp:31415
- Verify:
  adb forward --list

Run Drozer Console via Docker

- Start Drozer:
- docker run -it -p 31415:31415 drozerdocker/drozer console connect --server host.docker.internal

Test Connection

- Inside Drozer console run:
  run app.package.list: it will list the packages!

2.2 AndroGOAT Setup

AndroidGoat setup is just a straightforward android install:

adb install AndroGoat2.apk

3. Exploitation Scenarios

In this section we walkthrough the steps to perform a scan using the drozer utility on the androidgoat apk, this works as a initial recon that gives us the information about the target application's attack surface.

It enumerates activities, and services exposed by the app that can be further researched upon for weakness.

3.1 Drozer, JADX, AndroGOAT Exploitation PoC

1. Identify the Target Application

- List installed apps and filter for OWASP Goat:
  run app.package.list -f goat
- Note the **package name** (e.g., owasp.sat.agoat).

```
dz>
dz> run app.package.list -f goat
Attempting to run shell module
owasp.sat.agoat (AndroGoat - Insecure App (Kotlin))
dz>
dz>
dz>
```

2. Enumerate the Attack Surface

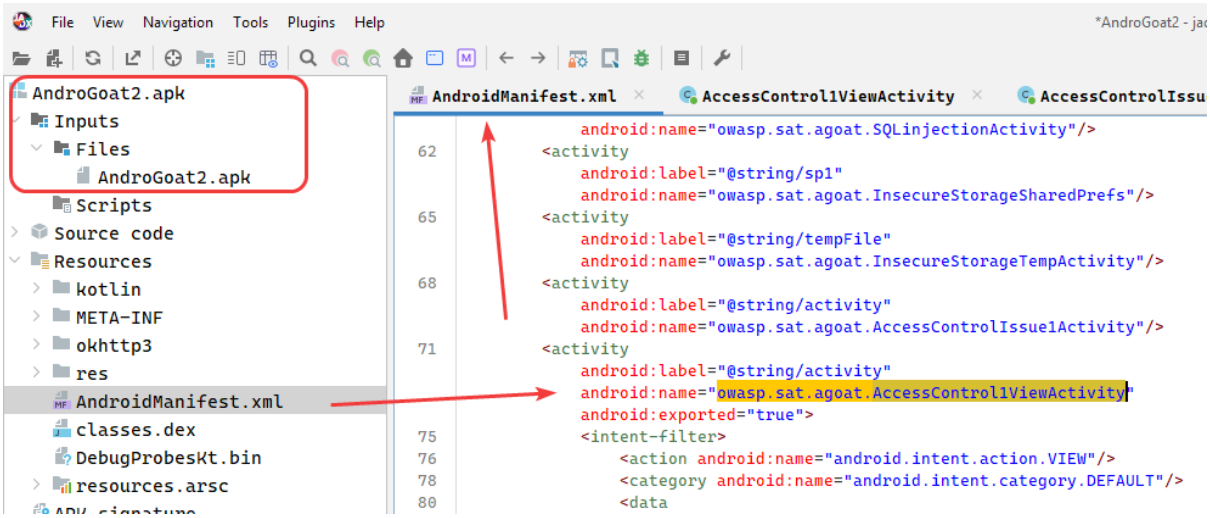
- Get exposed components:
  run app.package.attacksurface owasp.sat.agoat
- Enumerate activities in detail:
  run app.activity.info -a owasp.sat.agoat
- Identify any **exported activities** (i.e., activities accessible without authentication).

```
dz>
dz> run app.package.attacksurface -a owasp.sat.agoat
Attempting to run shell module
Exception occured: unrecognized arguments: -a
dz> run app.package.attacksurface owasp.sat.agoat
Attempting to run shell module
Attack Surface:
  2 activities exported
  2 broadcast receivers exported
  0 content providers exported
  1 services exported
    is debuggable
dz>
```

3. Perform Static Analysis Using JADX

- Open the androgoat APK in JADX GUI.

- Navigate to:
    - AndroidManifest.xml
    - Look for exported components, e.g.: android:name="owasp.sat.agoat.AccessControl1ViewActivity"
- Double-click the class to open its source code.



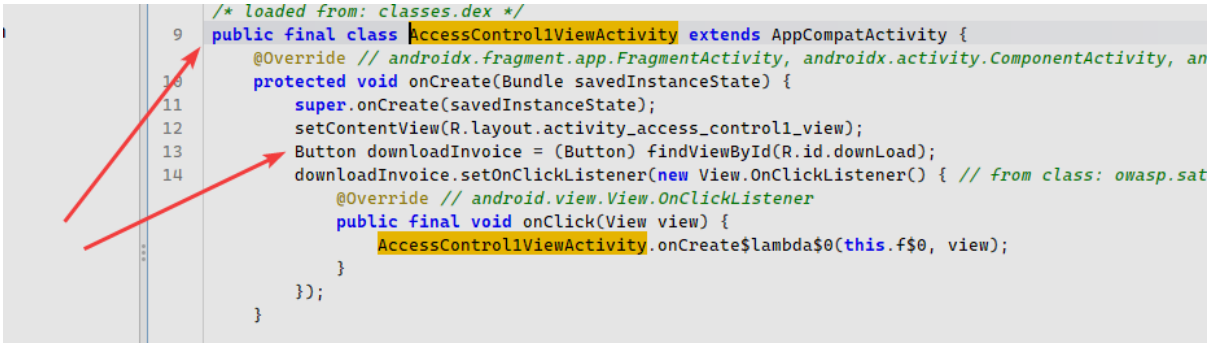
4. Review Activity Logic

- In AccessControl1ViewActivity, observe that onCreate() directly initializes buttons such as the **Download Invoice** button.
- No authentication check exists inside this activity's code.
- Example Observed Behaviour:
    - The onCreate() method sets the layout and registers a listener:

        Button downloadInvoice = findViewById(R.id.downLoad);

        downloadInvoice.setOnClickListener(...);

No validation is performed here.



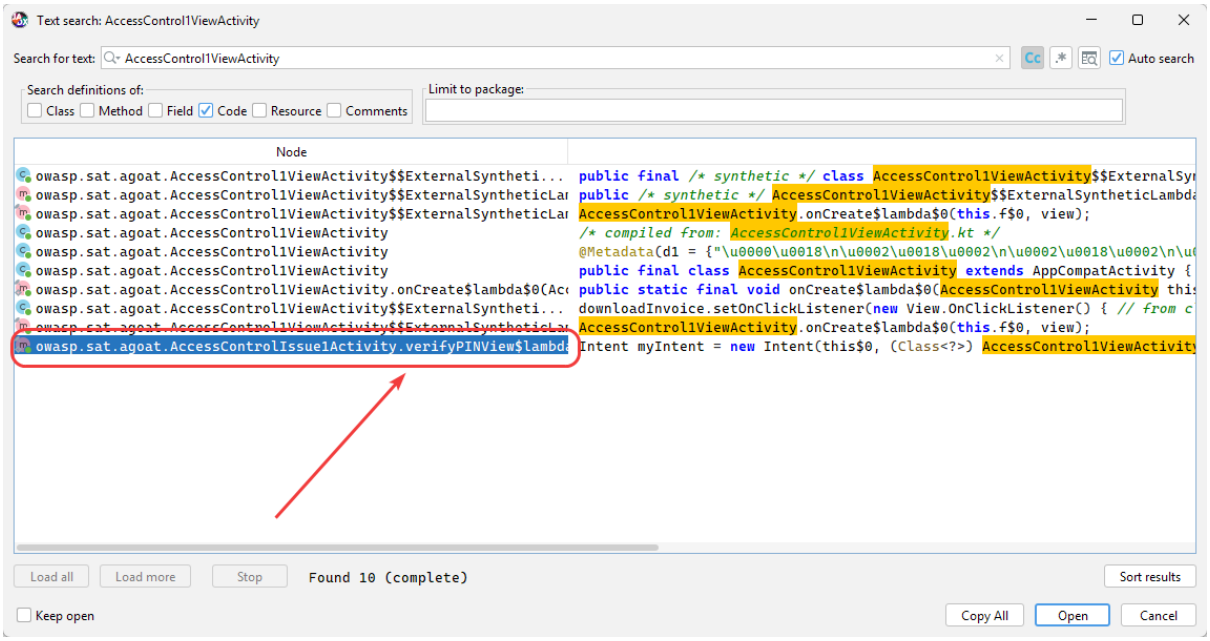
5. Trace the PIN Verification Flow

- Search for where PIN verification occurs (e.g., AccessControlIssue1Activity).
- Observe the method:

public static final void verifyPINView$lambda$1(...)

Logic:

  - If PIN is correct → show “PIN Verified” toast → launch AccessControl1ViewActivity.
  - If not → "Incorrect PIN".
- PIN verification happens outside the vulnerable activity.
- AccessControl1ViewActivity can run by itself without going through verifyPINView$lambda$1().



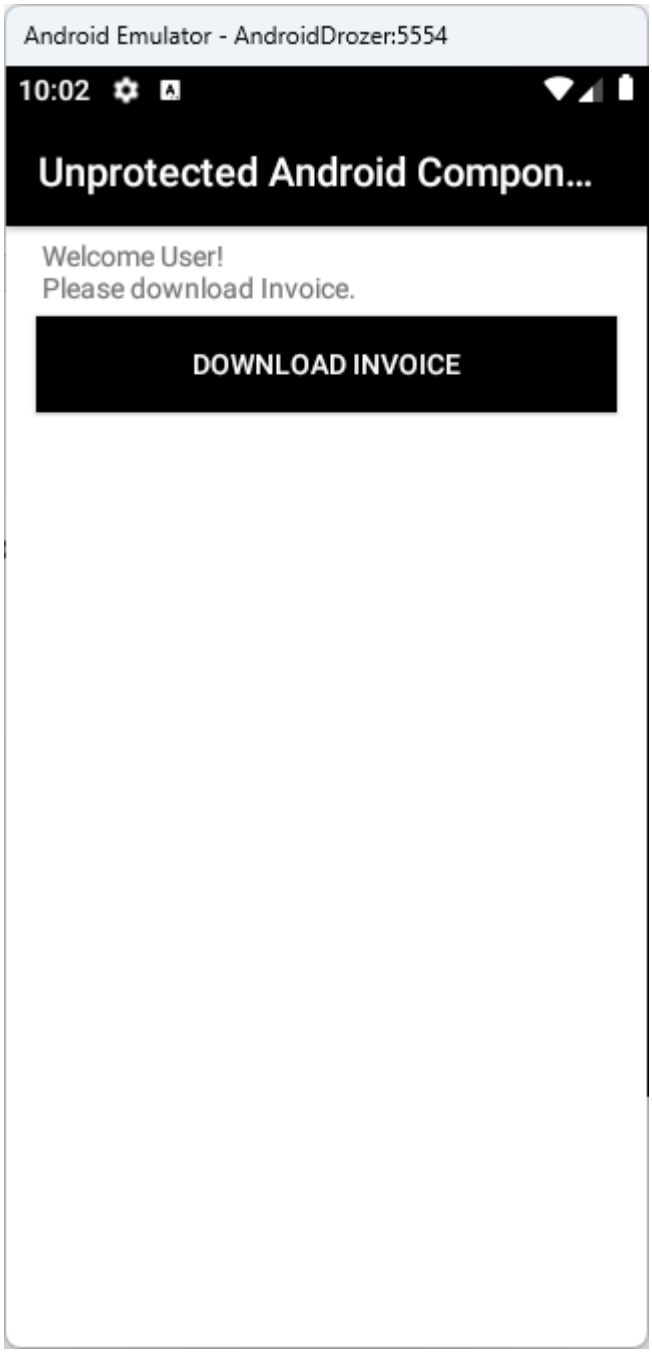
6. Exploit: Launch the Activity Directly Using Drozer

- Since the activity is exported, it can be launched directly:

- run app.activity.start --component owasp.sat.agoat owasp.sat.agoat.AccessControl1ViewActivity
- Result:
    - The protected activity opens without entering a PIN.
    - The invoice download button is immediately available.
    - This demonstrates broken access control / insecure activity export.

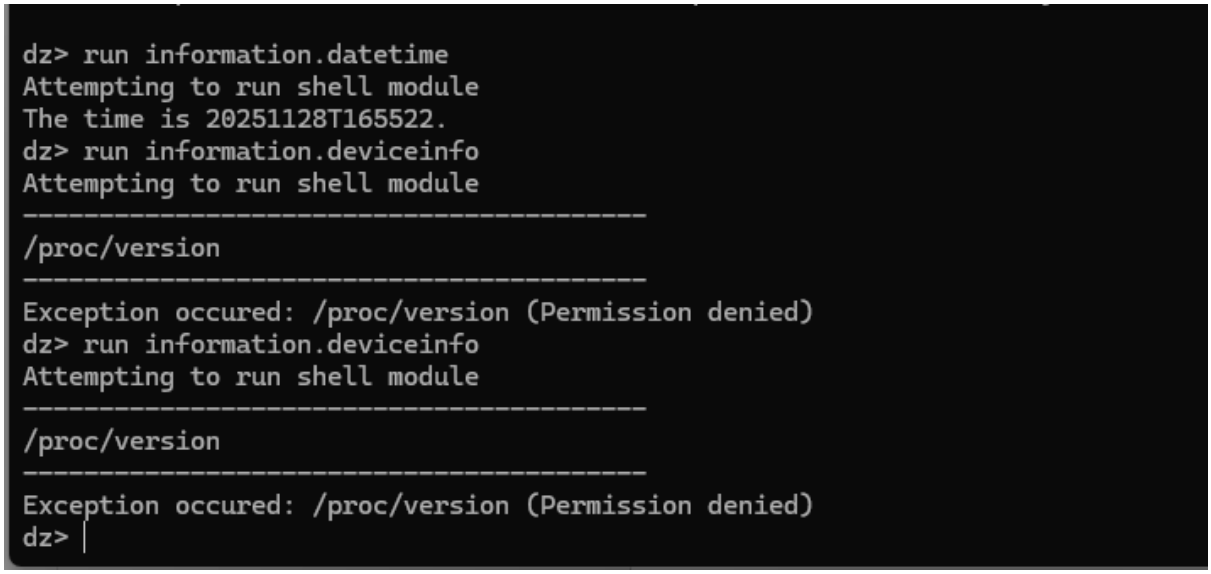
```
/* JADX INFO: Access modifiers changed from: private */
public static final void verifyPINView$lambda$1(AccessControlIssue1Activity this$0, View it) {
    Intrinsics.checkNotNullParameter(this$0, "this$0");
    EditText pinValue = (EditText) this$0.findViewById(R.id.pinValue);
    if (this$0.isPinCorrect(pinValue.getText().toString())) {
        Toast.makeText(this$0.getApplicationContext(), "PIN Verified", 1).show();
        Intent myIntent = new Intent(this$0, (Class<?>) AccessControl1ViewActivity.class);
        this$0.startActivity(myIntent);
        return;
    }
    Toast.makeText(this$0.getApplicationContext(), "Incorrect PIN entered", 1).show();
}
```

7. The activity allows document access without PIN verification.


Android Emulator - AndroidDrozer:5554
10:02
Unprotected Android Compon...
Welcome User!
Please download Invoice.
DOWNLOAD INVOICE

3.2 Drozer Recon on Prod Build

In this section, we performed a quick reconnaissance test on the Android production build using a Play Store-enabled emulator, and observed that none of the declared permissions appear as available at runtime.



3.3 Notepad App Assessment

A security assessment was conducted on the Android application Notepad Free using the drozer framework. The objective was to identify insecurely exposed components that could be leveraged for a security demonstration, such as unauthorized data access, component hijacking, or content provider exploitation.

The app was selected due to its wild popularity with over 10 million downloads and it looks like is maintained by a small dev team. This increases the chances of misconfigurations etc.

3.3.1.1 Attack Surface Mapping

```
dz> run app.package.attacksurface com.atomczak.notepat
Attempting to run shell module
Attack Surface:
  1 activities exported
  1 broadcast receivers exported
  1 content providers exported
  2 services exported
```

```
dz> run app.activity.info -a com.atomczak.notepat
Attempting to run shell module
Package: com.atomczak.notepat
```

com.atomczak.notepat.MainActivity
Permission: null

3.3.1.2 Content Provider Assessment

The manifest exposed some providers.

- Exported = true
- No read or write permissions defined
- Potentially accessible from any other app
- No path permissions defined (→ internal routing only)

This made the provider the strongest candidate for exploitation.

```
<provider
  name="com.atomczak.notepat.external.AtoDataProvider"
  exported="true"
  authorities="com.atomczak.notepat.AtoDataProvider">
</provider>
```

3.3.1.3 URI Discovery Attempts

Drozer's provider enumeration revealed multiple authorities

content://com.atomczak.notepat.AtoDataProvider/
content://com.atomczak.notepat.CachedFileProvider/
content://com.atomczak.notepat.com.squareup.picasso/

Attempts were made to:

A. Enumerate exposed paths using:

run app.provider.finduri com.atomczak.notepat

B. Query common provider paths to access files or databases.

Tried: /notes, /note, /items, /files, /data, /database, /attachments

All returned "Not implemented" or empty results, meaning:

- The provider exists
- But the root URI cannot be queried directly
- Internal paths are unknown or intentionally restricted

Further I ran the following drozer scanners:

scanner.provider.information

scanner.provider.injection
scanner.provider.traversal
scanner.provider.sqltables

The results of these scans did not return any exploitable avenues.

No accessible database tables
No SQL injection vectors
No directory traversal exposure
No MIME type leaks
No open read/write APIs

This indicates the provider is exported, but does not expose interactable data paths.

4. Results and Findings Summary

The security assessment using Drozer successfully demonstrated multiple weaknesses in the AndroGoat application, highlighting the risks posed by improperly secured Android components.

Through component enumeration and targeted exploitation, we were able to bypass the PIN-protected activity by directly invoking an exported component, effectively gaining unauthorized access to functionality that was intended to be restricted. Additionally, we leveraged Drozer's content-provider interaction modules to retrieve a document without authentication, confirming that sensitive files were exposed due to misconfigured or unprotected providers.

These findings collectively show how insecure activity exports, weak access controls, and exposed content providers can be exploited to compromise user data and application integrity.

The PoC validates the importance of enforcing strict component permissions, validating access control at every layer, and following OWASP MASVS guidelines to mitigate similar vulnerabilities in production applications.

5. Appendix

5.1 Work LOG

Date	Hours	Member	Description
17th Nov	2	Allwyn	Researching and setting up docker

18th Nov	2	Allwyn	Setting up the drozer console and agent APK
18th Nov	1	Allwyn	Setting up the Android emulator etc
20th Nov	2	Allwyn	Researching and learning about the Androgoat apk
21st Nov	2	Allwyn	Researching exploits, and reverse engineering with JadX
22nd Nov	3	Allwyn	Researching activity exploits on androgoat and recording the demo video.
23rd Nov	3	Allwyn	Final report writing, with screenshots, formatting, grammar, etc.
24th Nov	2	Allwyn	Notepad app recon and assessment using drozer

5.2 AI Use Table

AI Tool	Version and Account Type	Use Case Description
ChatGPT	4.0, 5.0 / Free account	I wrote the report sections in natural language and

		prompted AI for corrections, grammar, flow of language. All content, and ideas were mine. AI was used only as a tool to clarify ideas and rewrite at times for better readability.
ChatGPT	4.0, 5.0 / Free account	AI was used for exploit code explanations – the logic of the java code, why the exploit works, how the if-statements work, etc.

5.3 References

WithSecureLabs. *drozer*. GitHub. Accessed 2025. https://github.com/WithSecureLabs/drozer.

Drake, Joshua J., Zach Lanier, Collin Mulliner, Pau Oliva Fora, Stephen A. Ridley, and Georg Wicherski. *Android Hacker's Handbook*. Hoboken, NJ: Wiley, 2014.

Atomczak. "Notepad – Simple Notes." *Google Play Store*. https://play.google.com/store/apps/details?id=com.atomczak.notepat&hl=en.

Docker, Inc. "Install Docker Desktop on Windows." *Docker Documentation*. https://docs.docker.com/desktop/setup/install/windows-install/.

6. Chetan's Report

7. Abstract

This project investigates forensic artefacts extracted from Reddit and Telegram using Andriller CE and Android emulation. By analyzing Shared Preferences XML, Firebase tokens, Google Sign-In data, OAuth sessions, and event logs, the study demonstrates that meaningful identity and activity evidence remains recoverable even when message content is encrypted or inaccessible.

8. Background Research

Mobile application forensics has evolved significantly, as modern applications increasingly shift toward encrypted cloud-based storage and protected filesystem environments. Despite these protections, research consistently shows that metadata such as configuration files, login tokens, analytics logs, Firebase identifiers, and XML-based preferences remains locally accessible. These artefacts offer a rich source of evidence for investigators trying to reconstruct a user's identity, app usage, and device attribution. Tools such as Andriller CE, ALEAPP, ADB backup utilities, and manual XML/JSON parsing are commonly used for retrieving this type of metadata from Android applications.

Reddit and Telegram are widely used communication platforms that store critical metadata differently. Telegram employs encryption, which prevents access to message content on non-rooted devices. Reddit relies on OAuth authentication, Google Sign-In integration, Firebase Cloud Messaging, and analytics tracking, all of which produce identifiable artefacts stored in Shared Preferences files. These artefacts are crucial in forensic contexts because they provide strong connections between a user, an account, and a device—even without access to message bodies.

The purpose of this project is to analyze what forensic artefacts can be extracted from Reddit and Telegram installed on an Android emulator, and to understand how metadata can still provide evidence of app installation, login activity, profile identity, and backend communication.

9. Forensic Plan

To carry out the examination, the project used Andriller CE as the primary forensic extraction tool. Andriller performs logical acquisitions via ADB, retrieving SharedPreferences XML files, authentication tokens, installation identifiers, and additional configuration artefacts. This was supported by ADB utilities to install and run APKs, generate emulator activity, and dump application folders.

The forensic workflow followed these steps:

- Create an Android 13 emulator and install Reddit and Telegram.
- Perform login activities using valid accounts (Google Sign-In for Reddit, phone number login for Telegram).
- Generate user activity such as browsing posts, viewing screens, opening menus, and navigating through the apps.
- Use Andriller CE to extract logical backups of each app's data directory.
- Parse the extracted XML, JSON, and log files, focusing on Google Sign-In tokens, Reddit OAuth tokens, Firebase registration, analytics logs, and installation metadata.
- Interpret each artefact to determine its forensic significance, identity linkage, and timestamp relevance.

Most modern Android applications place sensitive data such as message content or chat logs into encrypted containers or rely exclusively on cloud-based storage. However, metadata structures like Shared Preferences XML often contain crucial remnants of authentication and identity information. For this reason, the underlying architecture of the apps guided the investigation toward metadata-driven forensics instead of decrypted content retrieval.

10. Detailed Analysis

10.1 Reddit Forensic Artefacts

Reddit proved to be the richest application for forensic extraction, as multiple Shared Preferences XML files revealed a variety of identifiers that confirm installation, login, activity, and identity linkage.

10.1.1 Firebase Cloud Messaging Token

One XML file contained Firebase Cloud Messaging (FCM) token data:

- Device's unique FCM token
- Token creation timestamp
- App version number
- Backend registration metadata

This confirmed that Reddit was **installed**, **executed**, and successfully **registered with Firebase**, proving first launch and backend connectivity.

10.1.2 Google Sign-In Artefact

The Google Sign-In JSON object stored in Shared Preferences contained:

- The user's Google email
- Google account ID (sub)
- OAuth scopes (email, openid)
- The Google ID token (JWT)
- Issued-at and expiration timestamps
- The server-side Client ID

This artefact alone creates a strong forensic link between the emulator, the user's identity, and the Reddit login session.

10.1.3 Reddit OAuth Authentication Token

A separate XML file contained:

- Reddit username (Ck___786)
- Token expiration timestamp
- Full Reddit OAuth session token
- Unique user IDs (aid, lid, rcid)

This evidence proves the presence of an **active authenticated Reddit session** on the device at the time of extraction.

10.1.4 Reddit Analytics Event Logs

Reddit stores extensive analytics logs (analytics_v2) that record:

- The posts viewed
- Comments opened
- Session timestamps
- App screens visited
- Device model and OS
- Navigational patterns (post_detail, home, comment threads)

These logs allowed reconstruction of user activity timelines, even though no message or chat content was available.

11. Telegram Forensic Artefacts

Telegram exhibited limited recoverable data due to its end-to-end encryption architecture. Telegram stores message content in encrypted databases using MTProto, with decryption keys inaccessible on non-rooted devices. However, several useful artefacts were still recoverable:

- Installation traces in /Android/data/
- Temporary cache files (images, icons, profile thumbnails)
- Login metadata
- Basic configuration XML files
- Network connectivity logs

Although message content was not accessible, these findings still confirm installation, account login, and general usage.

12. Results and Insights

The results clearly show that Reddit produces significantly more forensic artefacts than Telegram in non-rooted environments. Metadata extracted from Reddit provided a complete identity chain involving:

- Google account used to log in
- Reddit account username
- OAuth and FCM tokens
- Backend registration
- Analytics logs showing user behavior

These artefacts confirm installation, authentication, and active usage of the app with precise timestamps. Telegram, while more secure, still left sufficient metadata to prove installation, application launch, and image caching.

The project also revealed that **Reddit Chat does not function on emulators** because Reddit's backend disables chat features when Firebase and device integrity checks fail. This provides insight into the app's security mechanisms and explains why chat logs could not be produced.

12.1.1 Key insights gained

- Metadata-based forensics remains extremely powerful, even without decrypted chat databases.
- Google Sign-In artefacts provide strong personal attribution.
- OAuth tokens can confirm active login sessions.
- Firebase identifiers prove installation and backend communication.
- Analytics logs help reconstruct behavior and timeline.

Telegram's encryption prevents content retrieval, but installation proof is still possible.

12.1.2 Limitations encountered

- No root access → no Telegram message database decryption.
- Emulator environment restricted full filesystem visibility.
- Reddit chat was inaccessible due to firebase not supported.
- Some app versions required downgrading for analysis compatibility.

13. Conclusion

The forensic study demonstrates that even with the increasing use of encryption and cloud-based storage, mobile applications still retain numerous artefacts that can be used to prove identity, account usage, and application behavior. Reddit, in particular, stores extensive identity and activity data in Shared Preferences, making it highly valuable for forensic investigations. Although Telegram protects message content, metadata such as installation traces and login identifiers still provide insight into user actions. Future work could involve performing full filesystem extractions on rooted devices, analyzing encrypted databases by retrieving encryption keys, and performing network traffic captures to complement local artefact analysis.

14. Work Logs (Chetan Kaur)

Date	Hours	Detailed Description of Work Done
Nov 7, 2025	1.5	Set up Android emulator (Android 13 x86_64). Installed Android SDK components and ensured adb was functioning. Attempted Telegram installation on API 28 to test chat compatibility.
Nov 8, 2025	2	Installed Reddit APK and logged in using Google Sign-In. Conducted initial browsing to generate user activity (posts, comments, screen views). Explored app folders using Android Device Explorer.
Nov 9, 2025	1.5	Performed ADB backup (adb backup -apk -shared -all) for Telegram and Reddit. Identified limitations due to modern Android encryption. Tested multiple Telegram APK versions.
Nov 10, 2025	2	Ran Andriller CE for the first logical parsing of extracted backup. Analyzed SharedPreferences files for Reddit and interpreted recovered Firebase tokens and installation identifiers.
Nov 11, 2025	2	Conducted deeper analysis of Reddit artefacts: Google Sign-In metadata, OAuth token, user ID, account type, token expiration. Documented JSON and XML values for interpretation.
Nov 12, 2025	1.25	Navigated Reddit and generated multiple analytics events. Verified how Reddit logs user actions in analytics_v2. Captured details of post actions, session logs, and screen views.
Nov 13, 2025	2	Imported Andriller results into report folder. Interpreted Firebase Installation ID, Crashlytics ID, and backend communications evidence. Wrote evidence interpretation paragraphs.
Nov 14, 2025	1.5	Attempted Uber forensic extraction. Installed older Uber APK, generated map navigation activity, and tested whether ride data appears in local storage. Recorded findings and limitations.
Nov 15, 2025	1.25	Captured screenshots of emulator sessions, Andriller output, XML files, event logs, and settings files to include in the Detailed Analysis section.
Nov 16, 2025	1	Researched forensic literature for Shared Preferences artefact relevance. Compared findings against known forensic extraction limitations on Android 13+.
Nov 17, 2025	1.5	Drafted Background Research and Forensic Scenario sections. Wrote technical descriptions of how Andriller CE operates and its extraction workflow.
Nov 18, 2025	1.25	Structured analysis notes for Telegram, Uber, and Reddit. Documented both successful extractions and failed attempts. Developed conclusions and insights.

15. AI Use Section

This report was drafted using ChatGPT (GPT-5.1 and GPT-4o) under a Plus subscription. AI assistance was used for polishing language, restructuring sections, and improving readability. The extraction, evidence collection, screenshot interpretation, and artefact analysis were conducted manually. Human effort was central to performing the technical steps and validating all findings.

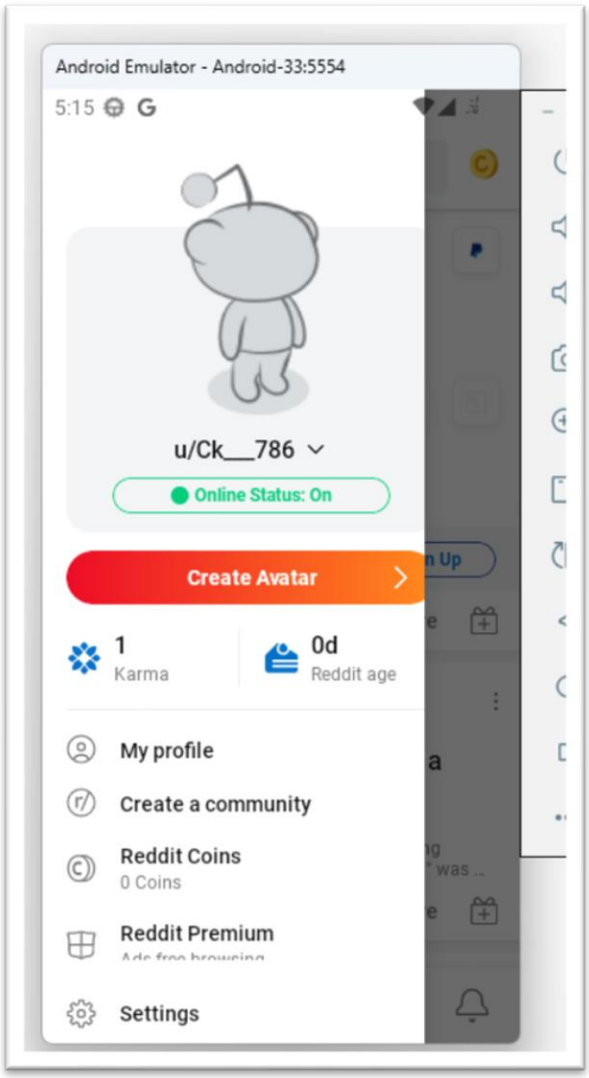
AI Tool Name	Account Type	Tasks for Which the AI Tool Was Used
ChatGPT (OpenAI GPT-5.1)	free	Used to interpret extracted artefacts from Reddit and Telegram. Generated forensic explanations for Shared Preferences tokens, Firebase IDs, Google Sign-In JSON, OAuth tokens, and analytics event logs. Helped structure sections: Background Research, Forensic Plan, Detailed Analysis, and Results.
ChatGPT (OpenAI GPT-4o)	free	Used to rewrite technical descriptions, improve clarity, and enhance professional academic tone. Assisted in summarizing complex artefacts, generating evidence interpretation paragraphs, and refining work-log entries.

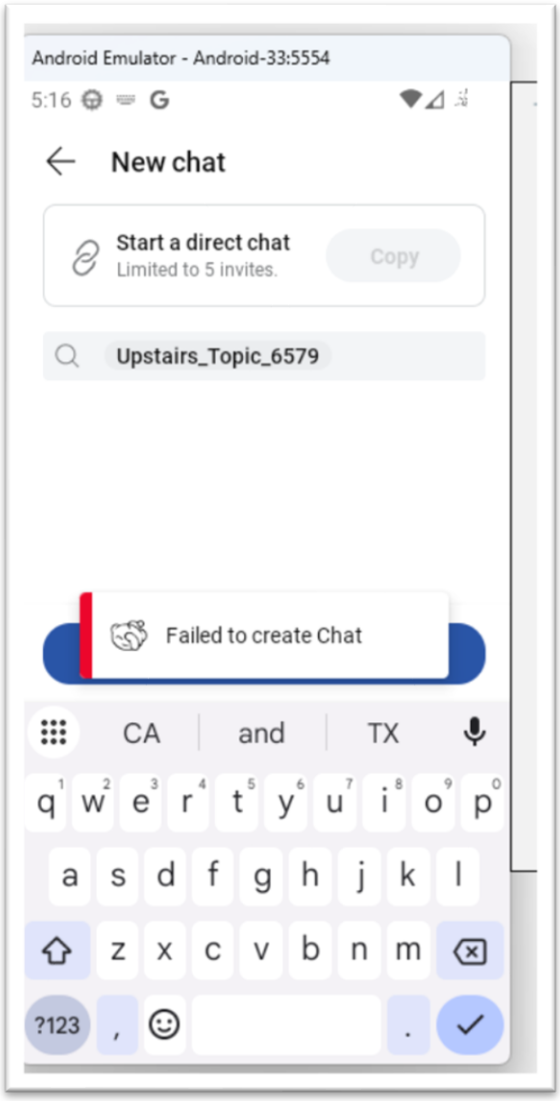
16. Appendix (Chetan Kaur)

16.1 Reddit analysis

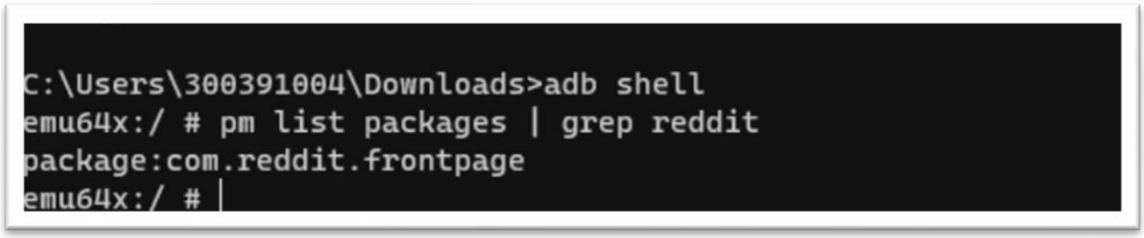
Installed the Reddit app on the emulator and created an account using Google sign-in and created a dummy account to try to chat. But chat failed because:

- Reddit Chat requires a **Firebase push token**
- Emulators do NOT provide a **real device ID**
- Reddit's backend detects "emulator → disable chat module."
- Only Feed, Posts, and Comments work
- The Chat tab will be **hidden** or **non-functional**





Checked the package for reddit:

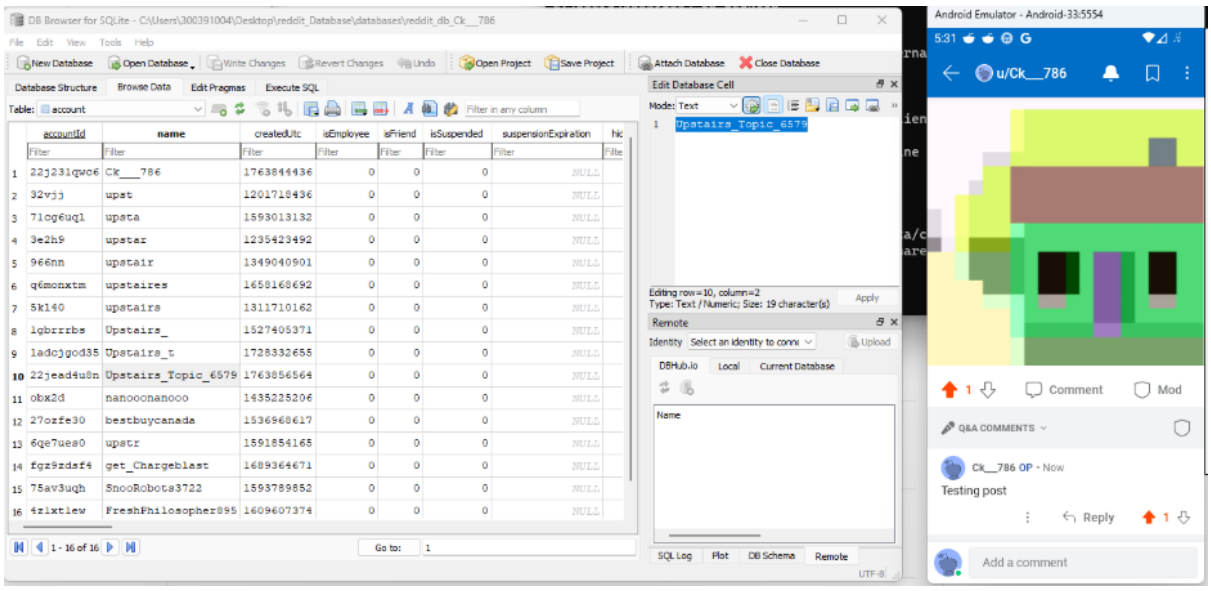


Analysis the databases in packages :

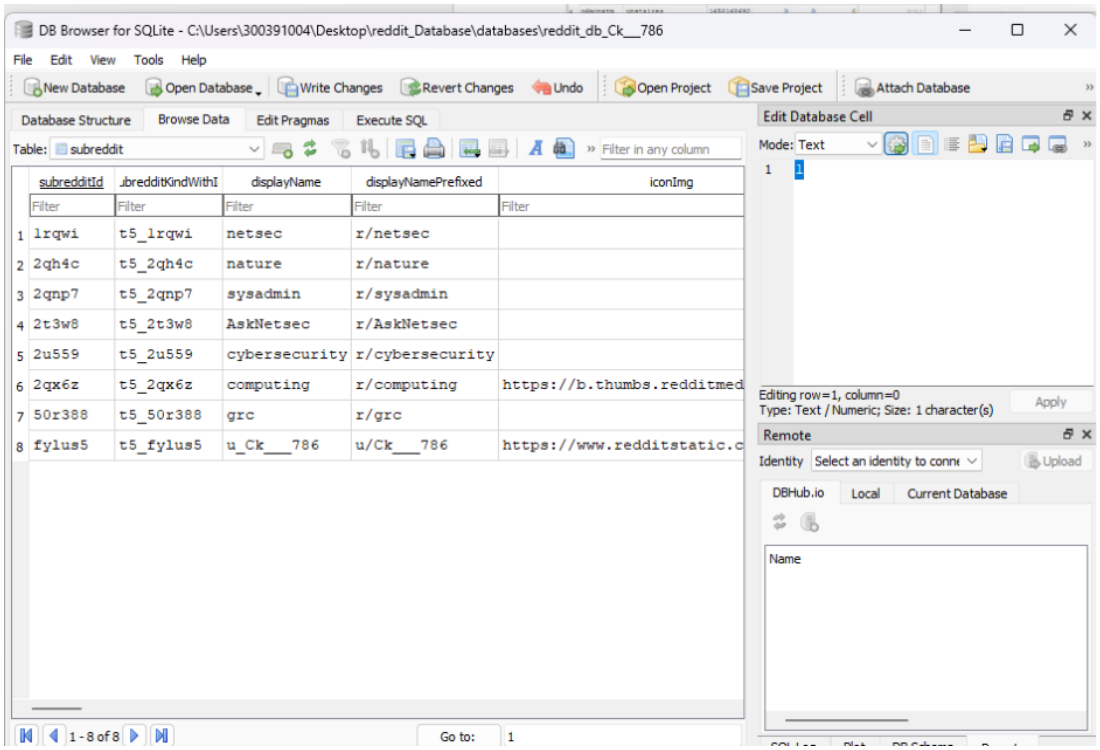
```
C:\Users\300391004\Downloads>adb shell
emu64x:/ # pm list packages | grep reddit
package:com.reddit.frontpage
emu64x:/ # cd /data/data/com.reddit.frontpage
emu64x:/data/data/com.reddit.frontpage # ls
app_dualcacheProviderStateCache  app_webview  code_cache  files      shared_prefs
app_textures                     cache        databases   no_backup
emu64x:/data/data/com.reddit.frontpage # cd databases/
emu64x:/data/data/com.reddit.frontpage/databases # ls
analytics_v2                                  reddit_db_Ck___786-shm
analytics_v2-shm                              reddit_db_Ck___786-wal
analytics_v2-wal                              reddit_db_anonymous
com.google.android.datatransport.events       reddit_user_Ck___786.db
com.google.android.datatransport.events-journal  reddit_user_Ck___786.db-journal
exoplayer_internal.db                         reddit_user_anonymous.db
exoplayer_internal.db-journal                 reddit_user_anonymous.db-journal
google_app_measurement_local.db               wallet_db
google_app_measurement_local.db-journal       wallet_db-shm
reddit_db_Ck___786                            wallet_db-wal
emu64x:/data/data/com.reddit.frontpage/databases #
```

Did the adb pull for reddit packages and analysis the reddit_db_Ck__786 Database and found that in account table all the keywords that I have reached for. As you can see in screenshot. As account Upstairs_Topic_6579 was my dummy account.

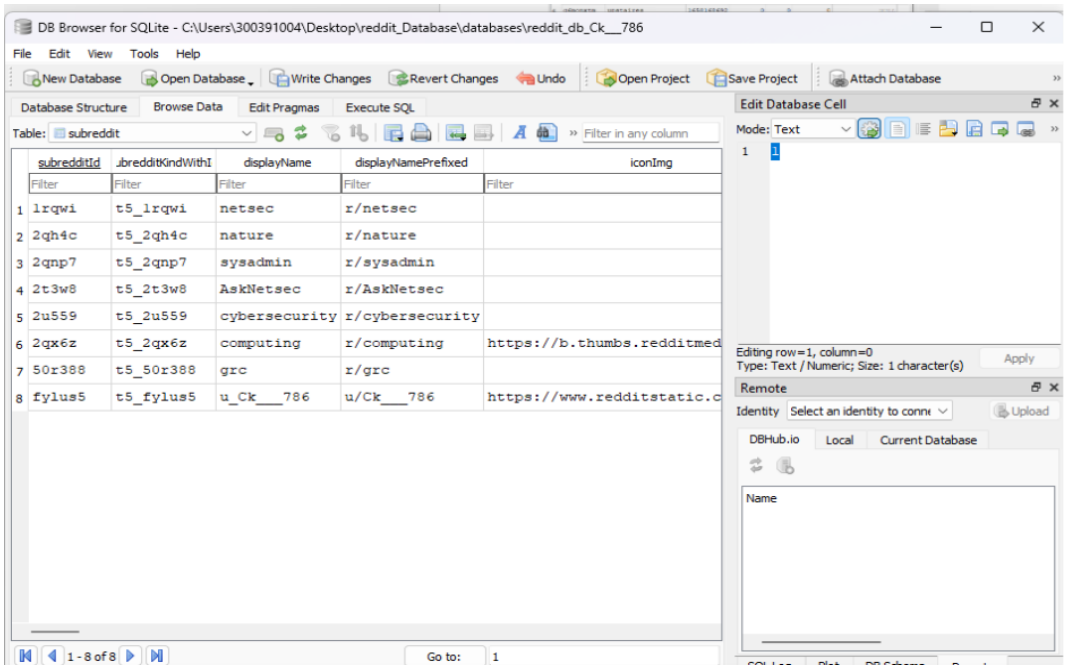
```
C:\Users\300391004\Desktop>adb root
adbd is already running as root

C:\Users\300391004\Desktop>adb pull /data/data/com.reddit.frontpage reddit_Database
adb: error: cannot create 'reddit_Database\shared_prefs\frc_1:933360113277:android:2918df7f0aab10ef_fireperf_settings.xml': No such file or directory

C:\Users\300391004\Desktop>adb pull /sdcard/Android/data/com.reddit.frontpage
/sdcard/Android/data/com.reddit.frontpage/: 57 files pulled, 0 skipped. 10.6 MB/s (3576099 bytes in 0.321s)
```



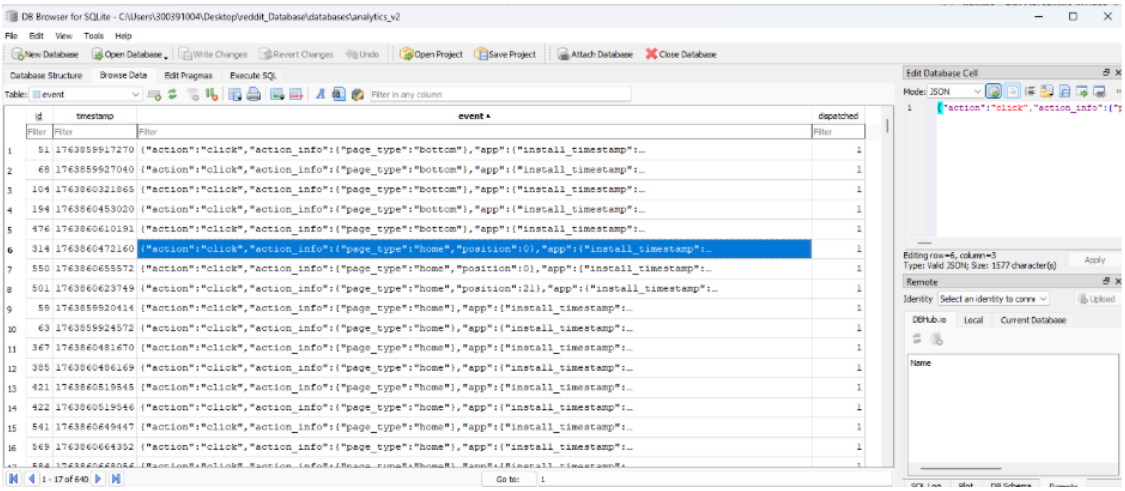
All the communities that is follow on reddit are listed under the subreddit table which provide all the details of the community I follow .



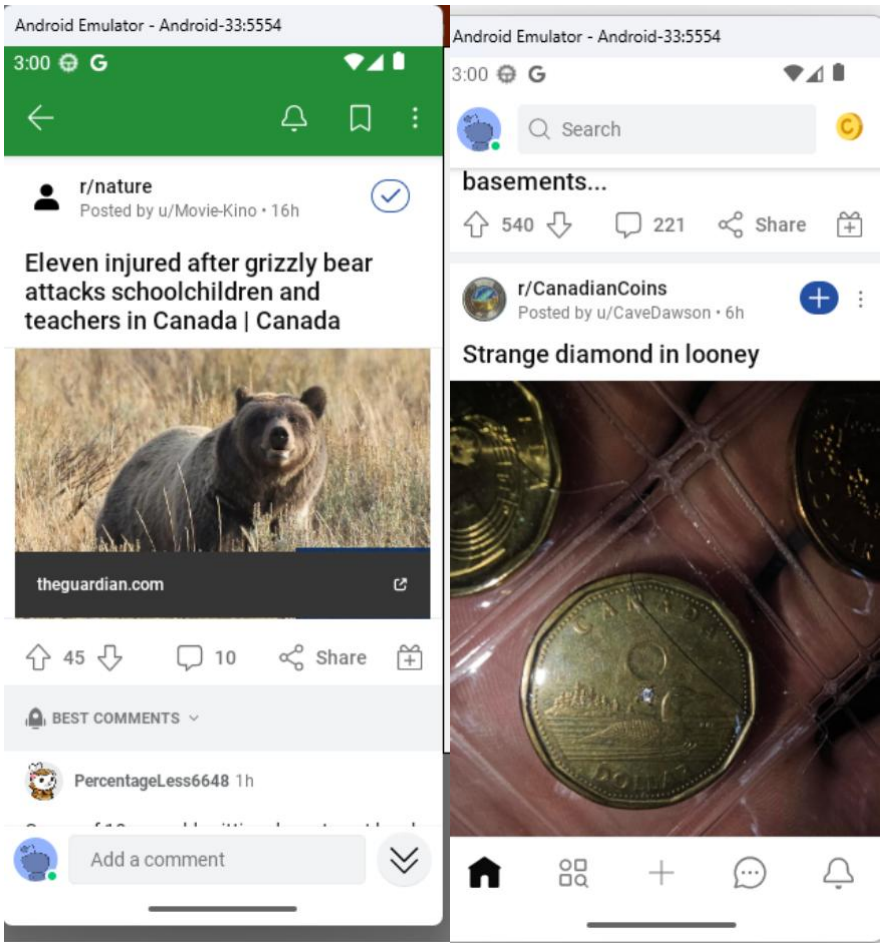
Similarly, all the searched keywords or ids are logged in query table.

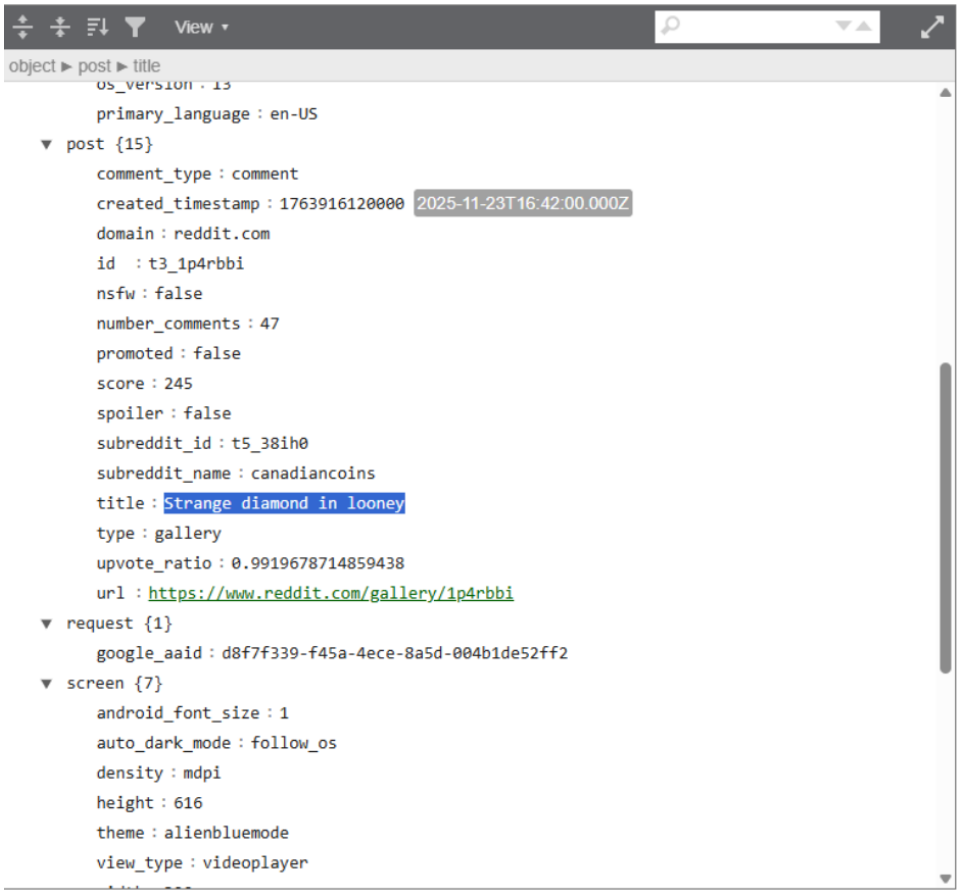


Further, analyzed the analytics_v2 database. As this database has a table called event, which has the details of the event you perform on Reddit, you can view the other posts, comments, and your own posts. It records all the activity any user does in the event table

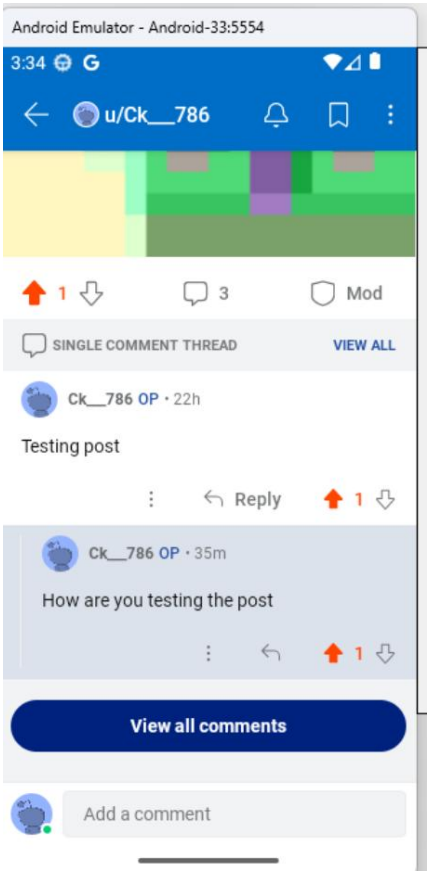


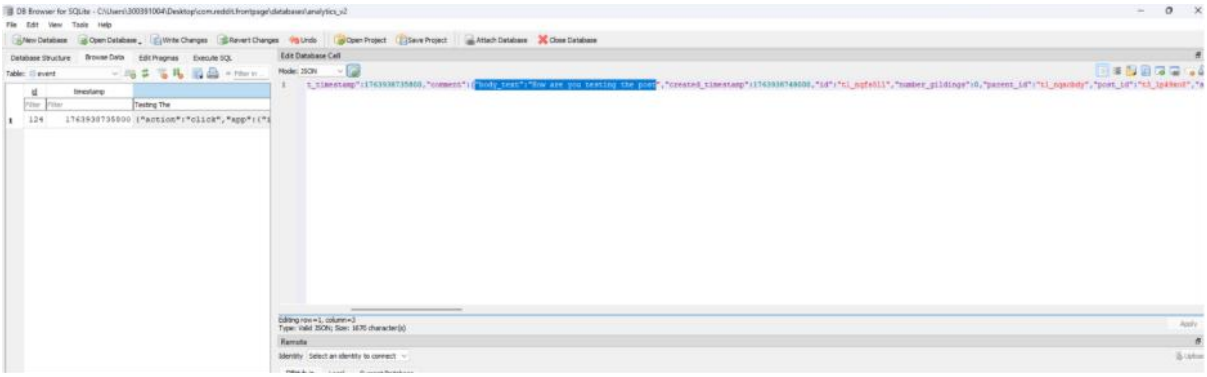
As I was viewing these posts and it was recorded in event table:





And the comments on the post I did are also recorded in the event table.



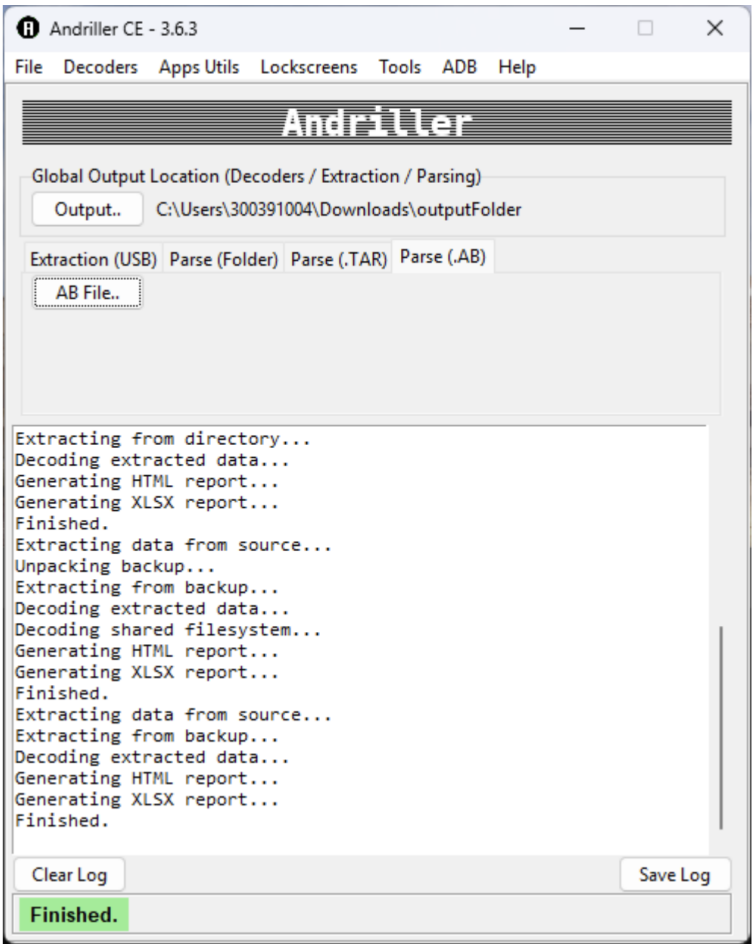


16.2 Extraction **of data using the Andriller tool**

Took the logical backup of the emulator.

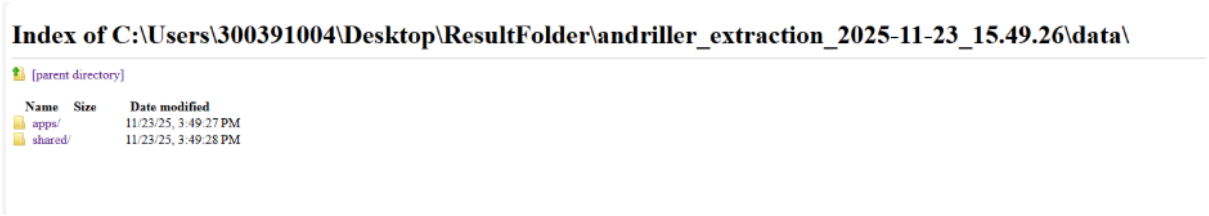
```
C:\Users\300391004\Desktop>adb backup -f backup.ab -apk -shared -all
WARNING: adb backup is deprecated and may be removed in a future release
Now unlock your device and confirm the backup operation...

C:\Users\300391004\Desktop>
```

After running the andriller, on the parse(.AB) button and uploading the logical backup on the andriller and it has created the report.

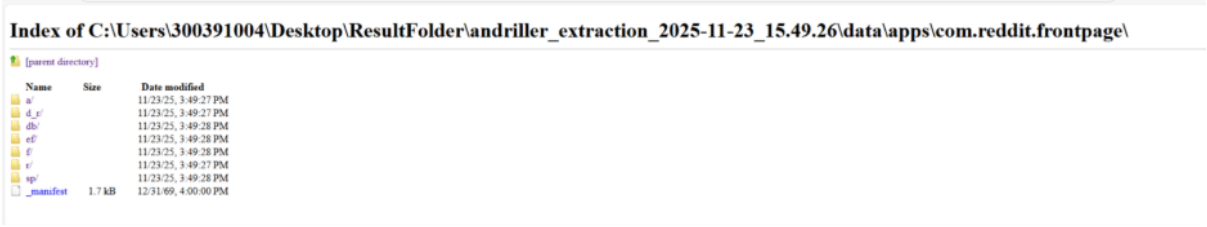


It has created the report in the output folder and given the HTML report to show the extracted data in a more presentable format.

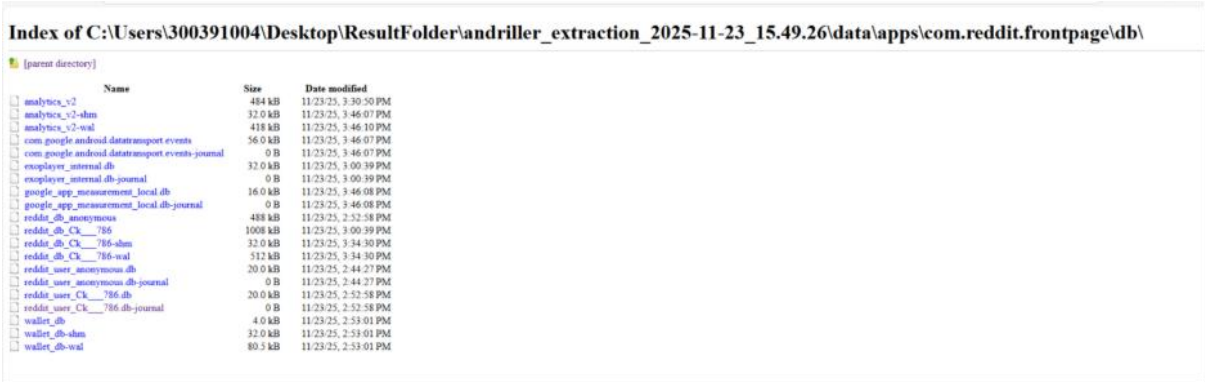
And I went through the apps folder on the Android report and saw the package created for the Reddit app.



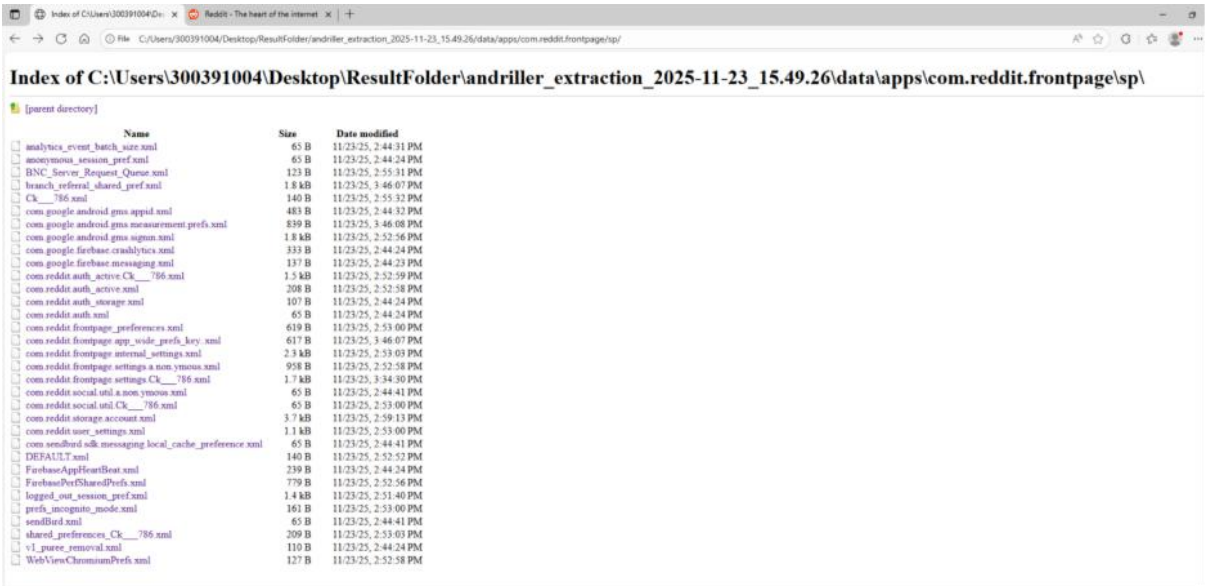
All the tables are listed in the reddit folder created by the andriller, even it gives all the hidden databases in well organised structure. (in screenshots)



To analyse the databases have to download the db.



In the shared preference folder, there were some tokens and IDs that I observed, and in the databases, the Google ID signed in was hidden. but on shared preferences, I can view that details.



Index of C:\Users\300391004\Desktop\ResultFolder\andriller_extraction_2025-11-23_15.49.26\data\apps\com.reddit.frontpage\sp\

[parent directory]

Name	Size	Date modified
analytics_event_batch_size.xml	65 B	11/23/25, 2:44:31 PM
anonymous_session_pref.xml	65 B	11/23/25, 2:44:24 PM
BNC_Server_Request_Queue.xml	123 B	11/23/25, 2:55:31 PM
branch_referral_shared_pref.xml	1.8 kB	11/23/25, 3:46:07 PM
Ck___786.xml	140 B	11/23/25, 2:55:32 PM
com.google.android.gms.appid.xml	483 B	11/23/25, 2:44:32 PM
com.google.android.gms.measurement.prefs.xml	839 B	11/23/25, 3:46:08 PM
com.google.android.gms.signin.xml	1.8 kB	11/23/25, 2:52:56 PM
com.google.firebase.crashlytics.xml	333 B	11/23/25, 2:44:24 PM
com.google.firebase.messaging.xml	137 B	11/23/25, 2:44:23 PM
com.reddit.auth_active.Ck___786.xml	1.5 kB	11/23/25, 2:52:59 PM
com.reddit.auth_active.xml	208 B	11/23/25, 2:52:58 PM
com.reddit.auth_storage.xml	107 B	11/23/25, 2:44:24 PM
com.reddit.auth.xml	65 B	11/23/25, 2:44:24 PM
com.reddit.frontpage_preferences.xml	619 B	11/23/25, 2:53:00 PM
com.reddit.frontpage.app_wide_prefs_key.xml	617 B	11/23/25, 3:46:07 PM
com.reddit.frontpage.internal_settings.xml	2.3 kB	11/23/25, 2:53:03 PM
com.reddit.frontpage.settings.a.non.ymous.xml	958 B	11/23/25, 2:52:58 PM
com.reddit.frontpage.settings.Ck___786.xml	1.7 kB	11/23/25, 3:34:30 PM
com.reddit.social.util.a.non.ymous.xml	65 B	11/23/25, 2:44:41 PM
com.reddit.social.util.Ck___786.xml	65 B	11/23/25, 2:53:00 PM
com.reddit.storage.account.xml	3.7 kB	11/23/25, 2:59:13 PM
com.reddit.user_settings.xml	1.1 kB	11/23/25, 2:53:00 PM
com.sendbird.sdk.messaging.local_cache_preference.xml	65 B	11/23/25, 2:44:41 PM
DEFAULT.xml	140 B	11/23/25, 2:52:52 PM
FirebaseAppHeartBeat.xml	239 B	11/23/25, 2:44:24 PM
FirebaseSetSharedPrefs.xml	779 B	11/23/25, 2:52:56 PM
logged_out_session_pref.xml	1.4 kB	11/23/25, 2:51:40 PM
prefs_incognito_mode.xml	161 B	11/23/25, 2:53:00 PM
sendBird.xml	65 B	11/23/25, 2:44:41 PM
shared_prefrences_Ck___786.xml	209 B	11/23/25, 2:53:03 PM
v1_puree_removal.xml	110 B	11/23/25, 2:44:24 PM
WebViewChromiumPrefs.xml	127 B	11/23/25, 2:52:58 PM

I recovered internal configuration artefacts such as Firebase Cloud Messaging (FCM) tokens stored inside the app's SharedPreferences. These artefacts prove that the app was installed, initialized push notifications, and communicated with Google's backend. Timestamps found inside the XML file provide evidence of the exact time when the token was created or refreshed.

This is **legitimate digital evidence** because it shows:

- App installation
- App execution
- Timestamped activity
- Backend registration
- Device-specific identifiers

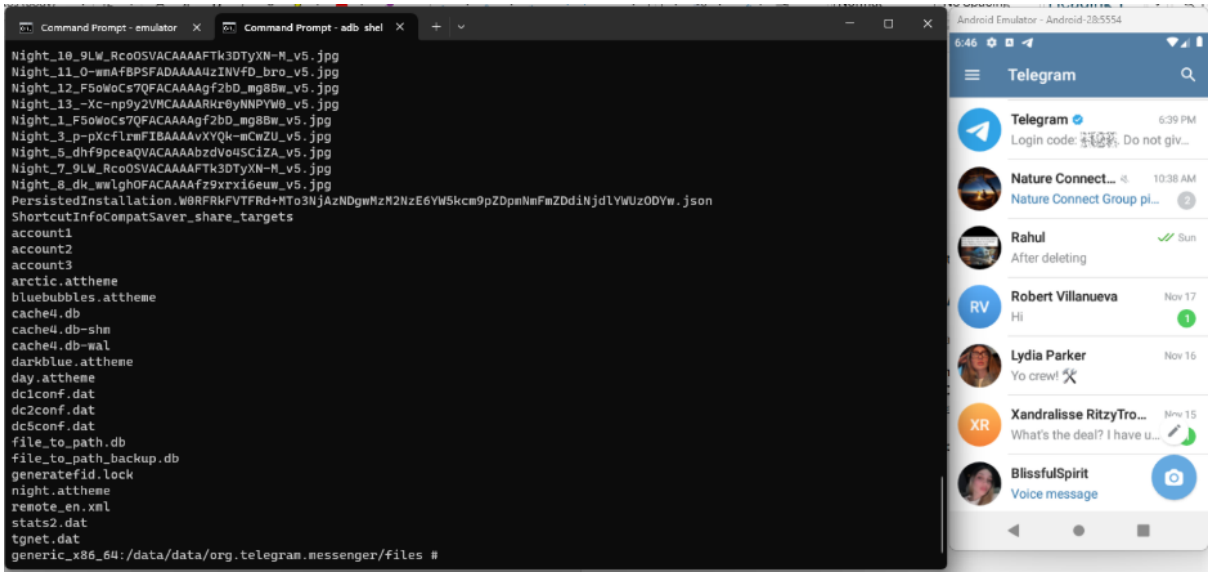

```
<map>
  <string
  name="W0RFRkFVTFRd+MTo5MzMzNjAxMTMyNzc6YW5kcm9pZDoyOTE4ZGY3ZjBhYWIxMGVm|S|cre">176
  3937863575</string>
  <string
  name="|T|933360113277|*">{"token":"cRllyyv0QJKMSw3gSiCsNZ:APA91bHMuDT0zGpQY3jlm_tnPRiaZr25yv
  l92itejBIxlHShLBXUrdEUHpZsTKGf8J9q5Hj-
  m6mqicPeO3QzYFViHYPeAn56NTe8h4Geto5O9bip1SIfJxs","appVersion":"426592","timestamp":176393787
  2278}</string>
</map>
```

16.3 Reddit Account Profile Artefact

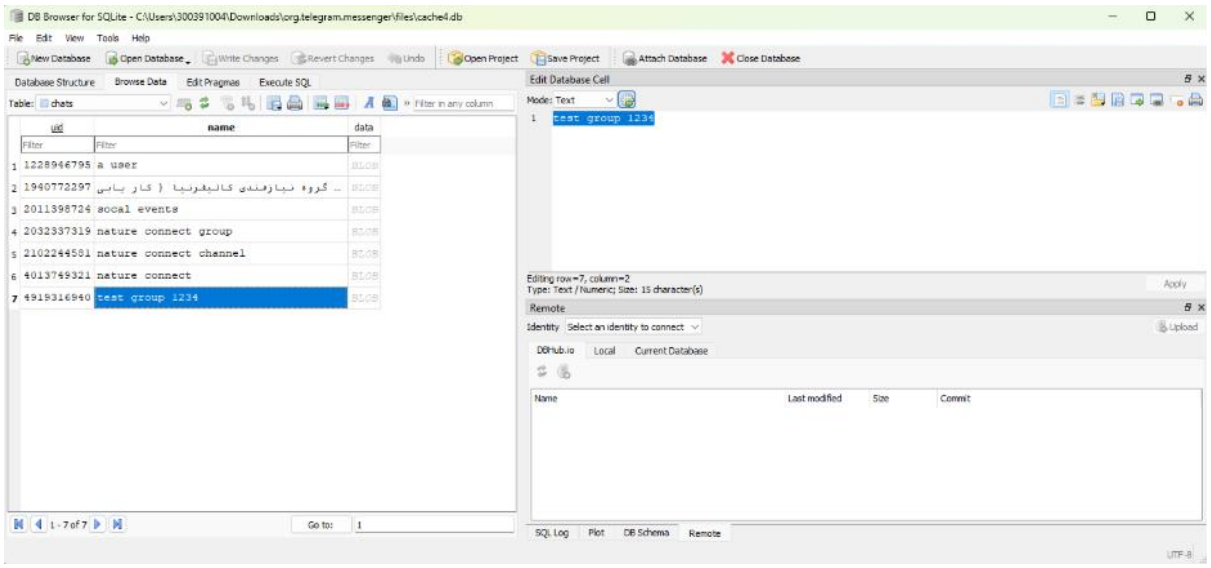
A detailed Reddit user-profile artefact was recovered from the application's SharedPreferences XML file. The entry contained a complete JSON object representing the authenticated Reddit account session on the device. This data included the Reddit username (*Ck___786*), the unique user identifier (*22j231qwc6*), the verified email address linked to the account, and profile-related metadata such as account creation time, karma scores, moderator status, inbox activity, profile description, and associated subreddit information. The presence of this artefact conclusively demonstrates that the Reddit application was logged into using this specific user account on the examined device. Additionally, the linked Google identity and verified email create a strong attribution link between the user and the Reddit account. This constitutes high-value forensic evidence indicating user identity, authentication activity, and application usage, as well as user-generated content stored locally on the device.

16.4 Telegram

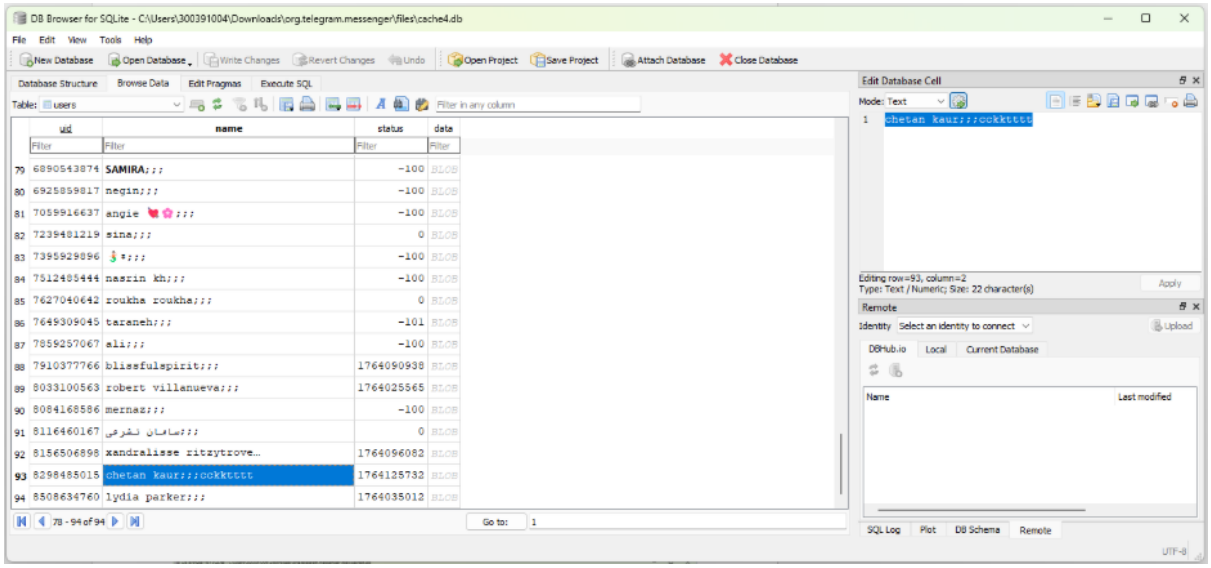
As Andriller is more efficient in extracting data from the SD Card, I tried to put all the data in a telegram package to save, and then did the logical backup and uploaded it to the Andriller, and got data to analyse, which telegram stored on the premises.



This screenshot shows the group I have joined on Telegram (cache4.db) under the files folder.



The list of users in the users table:



These messages are under the messages_v2 tables in blob.