Douglas College

Module Code:
CSIS 4440

Module Title:
Mobile Cybersecurity
Progress Report

Group: F25_4440_G10

Year & Semester:
2025 Fall

Student ID	Student Name
300391004	Chetan Kaur

Instructor: Priya

Due Date: 16 November 2025

Table of Content

Work Logs

Date	Student Name	Hours	Description of Work
10/29/2025	Chetan	2	Attempted to set up the Android SDK and command-line tools on macOS. Encountered multiple configuration issues and missing packages and concluded that meeting the project requirements on macOS would be complex.
11/02/2025	Chetan	3	Switched to a Windows laptop and continued Android environment setup. Installed Android tools and attempted to run Telegram on the emulator but faced login issues because the newer Telegram version did not provide an authentication code directly to the emulator.
11/04/2025	Chetan	3	Investigated Telegram login and authentication issues on the emulator. Researched version compatibility and documentation, then configured Telegram version 11.7.4 on Android 33. Successfully received the login code on the Telegram app installed on a physical cell phone and used it to sign in on the emulator.
11/06/2025	Chetan	3	Manually examined Telegram's databases on the emulator at /data/data/org.telegram.messenger/databases. Identified event and events-journal files and explored additional directories such as cache and files (including some JPG files). Used adb pull to copy these folders to the desktop for analysis.
11/07/2025	Chetan	2.5	Installed Andriller CE and set up the required Python environment on Windows. Verified that Andriller ran correctly and explored its main features and interface.

11/09/2025	Chetan	2.5	Pointed Andriller to the previously pulled Telegram data folder but found that no usable report was generated. Switched to performing a logical backup instead, uploaded the resulting. ab file into Andriller and obtained a Shared Storage report showing thumbnail-related entries (Movies, Pictures, Music, _database_uuid, size, and modified date). No meaningful Telegram application evidence was recovered in this report.
11/11/2025	Chetan	1.5	Attempted a physical-style backup of the Telegram data folder and produced a .dd image but confirmed that Andriller cannot directly process or analyze .dd extension files.
11/13/2025	Chetan	2	Reviewed a research article indicating that Andriller can extract additional artefacts such as Wi-Fi passwords, but primarily when used directly on a physical Android device rather than only on emulator data or standalone images. Reflected on how this limitation affects our project approach.
11/14/2025	Chetan	2	Tried to install and analyze the Uber application on the emulator. Encountered installation/runtime issues caused by CPU architecture mismatch between the Uber APK (ARM-only) and the x86 Android 33 emulator image.
11/15/2025	Chetan	2	Attempted to set up the Avilla forensic tool but found that it requires a VHD-based environment and sufficient disk space for partitions, which was not available. Reviewed documentation and YouTube tutorials for Avilla but decided to continue focusing on Andriller and exploring other applications instead.

Description of Work Done

So far, my work has focused on setting up a stable Android forensic environment, acquiring Telegram application data from an emulator, and evaluating how well Andriller can process that data. I began by attempting to configure the Android SDK and command-line tools on macOS but ran into multiple configuration problems and missing dependencies. Based on those challenges, I decided to move to a Windows laptop, where I reinstalled the Android tools and created an Android 33 emulator. I then installed Telegram and investigated authentication behaviour, eventually choosing Telegram version 11.7.4. Since the emulator did not receive the login code directly, I used my physical phone's Telegram app to obtain the verification code and successfully log in on the emulator, allowing me to generate test messages and activity for later analysis.

Once Telegram was running, I manually explored its internal data directories at /data/data/org.telegram.messenger/, including the databases, cache, and files folders. I identified database files such as event and events-journal, as well as image files in the files directory, and used adb pull to copy these directories to my desktop for further examination. After that, I installed Andriller CE, configured the required Python environment, and verified that the tool executed correctly. I first tried to load the pulled Telegram data folder directly into Andriller, but it did not generate a usable application-specific report. To work around this, I performed a logical backup, produced a "ab" file, and imported it into Andriller. This resulted in a Shared Storage report that listed thumbnail-related entries (e.g., _database_uuid in Movies/Pictures/Music folders) but did not reveal any meaningful Telegram chat or account artefacts.

I then experimented with a more physical-style backup of the data folder to produce a .dd file, only to confirm that Andriller does not natively support parsing raw .dd images. To better understand Andriller's strengths and limitations, I reviewed a research article showing that the tool can extract additional artefacts such as Wi-Fi passwords when used directly on a physical device, which helped explain why my emulator-based and image-based approaches were limited. I also attempted to broaden the app scope by installing Uber on the emulator, but the APK failed due to an ARM vs. x86 architecture mismatch. Finally, I explored the Avilla forensic tool and its documentation; however, Avilla requires a VHD-based environment and more disk space and partitioning than I currently have, so I decided to continue focusing on Andriller and additional applications that are more compatible with my setup.

My planned next steps are to generate clearer and more diverse artefacts using apps that work reliably on the emulator and produce data that Andriller can easily detect. I will use applications such as Reddit, Amazon Shopping, Flipkart, and Zomato, since they consistently create unencrypted cached images, thumbnails, JSON files, and other app-generated artefacts that appear in Andriller's Shared Storage and file extraction reports. After generating user activity in these apps, I will process both logical backups and ADB-pulled directories through Andriller to compare which artefacts are recoverable in each method.

As part of this work, I will also document the limitations encountered with Telegram and other encrypted apps, where modern security measures and emulator restrictions prevent meaningful extraction. Finally, I will integrate all findings screenshots, extracted paths, example artefacts, and tool limitations into the final project report.

AI Use Section

AI Tool Name, Version	Specific feature / task for which the AI tool was used
ChatGPT (OpenAI's GPT-5.1)	Asked for help understanding and troubleshooting Android emulator issues (ARM vs x86 architecture, AVD configuration, and adb backup commands).
ChatGPT (OpenAI's GPT-5.1)	Used to clarify the capabilities and limitations of Andriller CE (logical vs physical backups, ab vs .dd support and expected artefacts).
ChatGPT (OpenAI's GPT-5.1)	Requested explanations about Telegram data storage paths and database files (e.g., /data/data/org.telegram.messenger/databases) and their roles.

Value Added

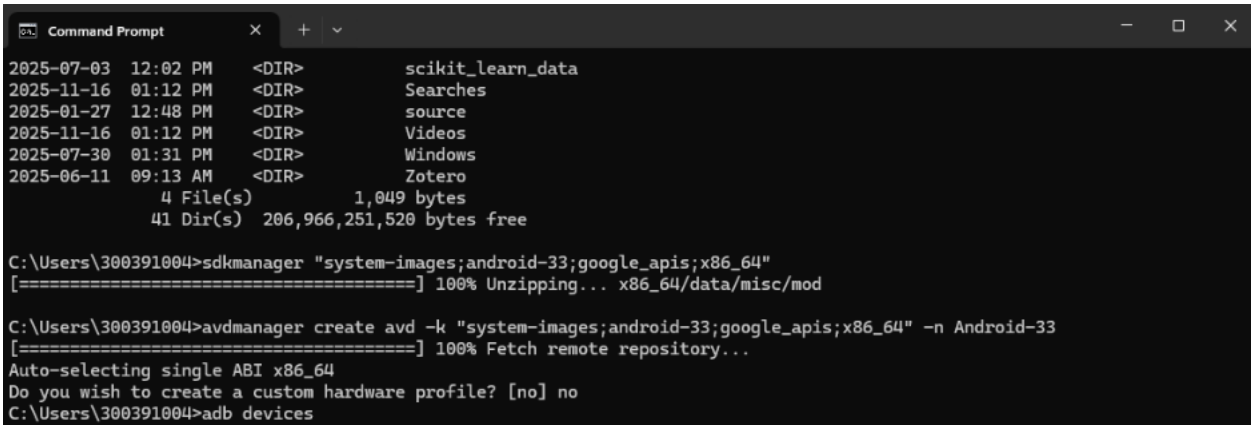
Although I used ChatGPT (OpenAI's GPT-5.1) to understand emulator issues, Andriller CE limitations, and Telegram data paths, I added value by installing and configuring the Android SDK and AVDs myself, running and adjusting the adb commands on my own system, and resolving errors. I manually located and pulled the Telegram database and files, tested different backup methods (logical, partial physical), confirmed what Andriller could and could not parse, and interpreted these findings in the context of our project goals and the research article I reviewed.

Appendix

SDK Tool Setup

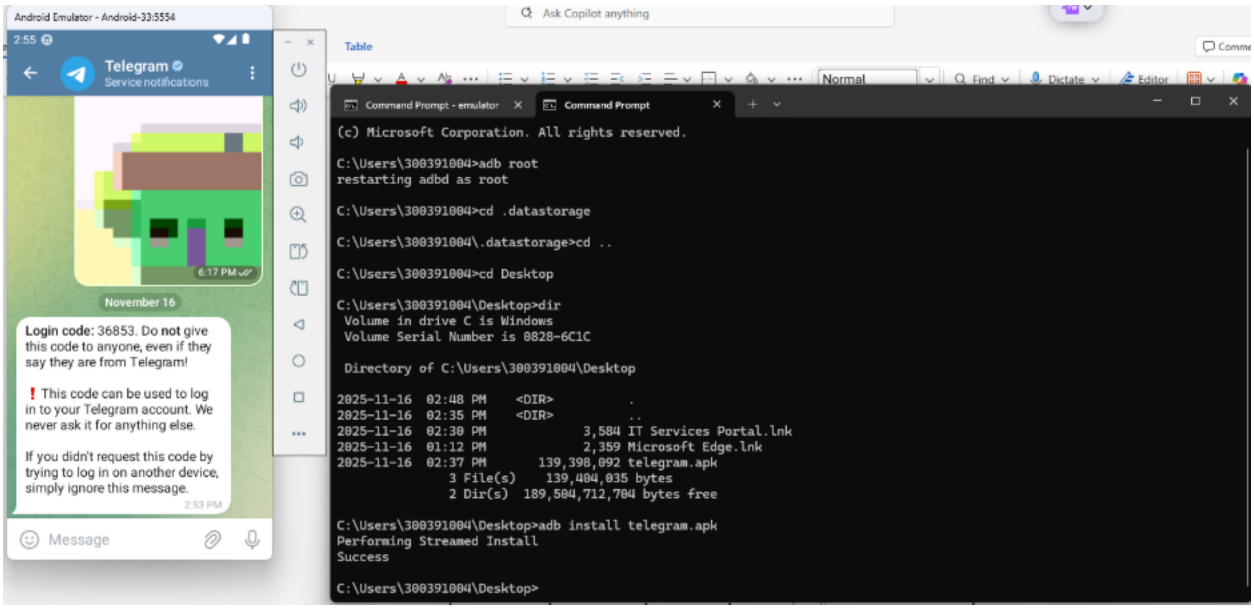
- Installed the required Android system image using:
- sdkmanager "system-images; android-35; google_apis;x86_64"
- Created an emulator with:

avdmanager create avd -k "system-images;android-33; google_apis;x86_64" -n Android-33



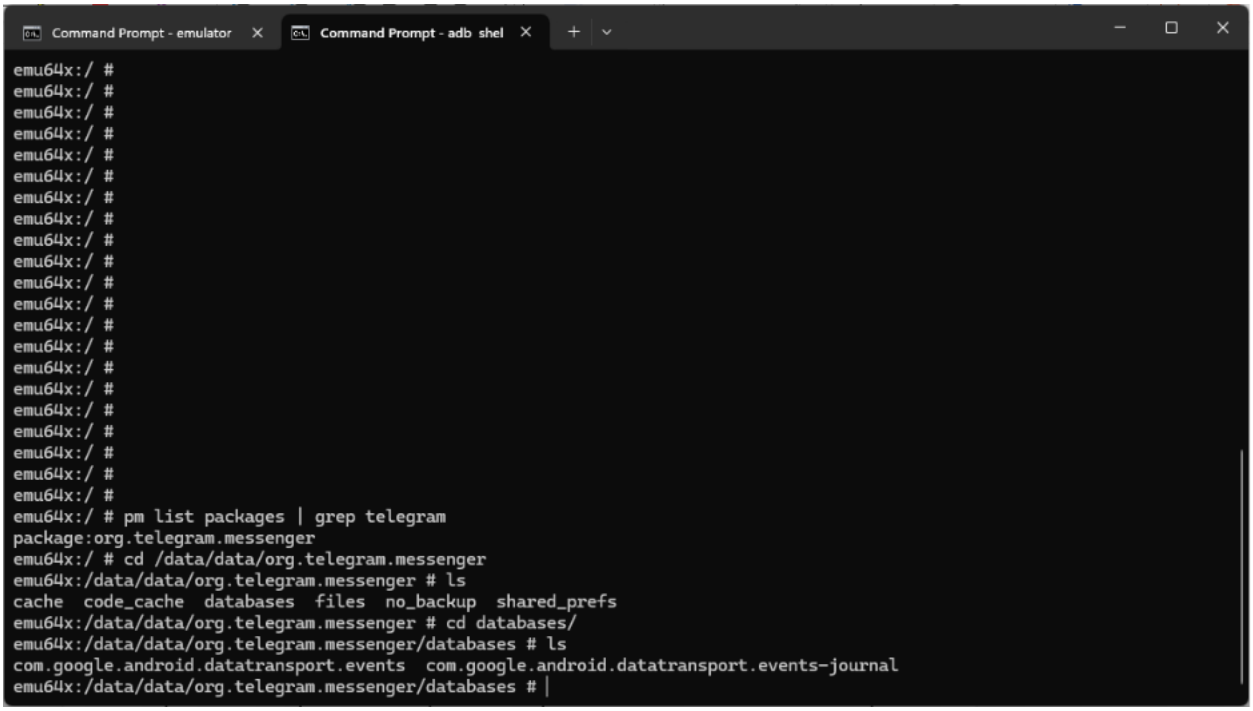
Telegram Setup

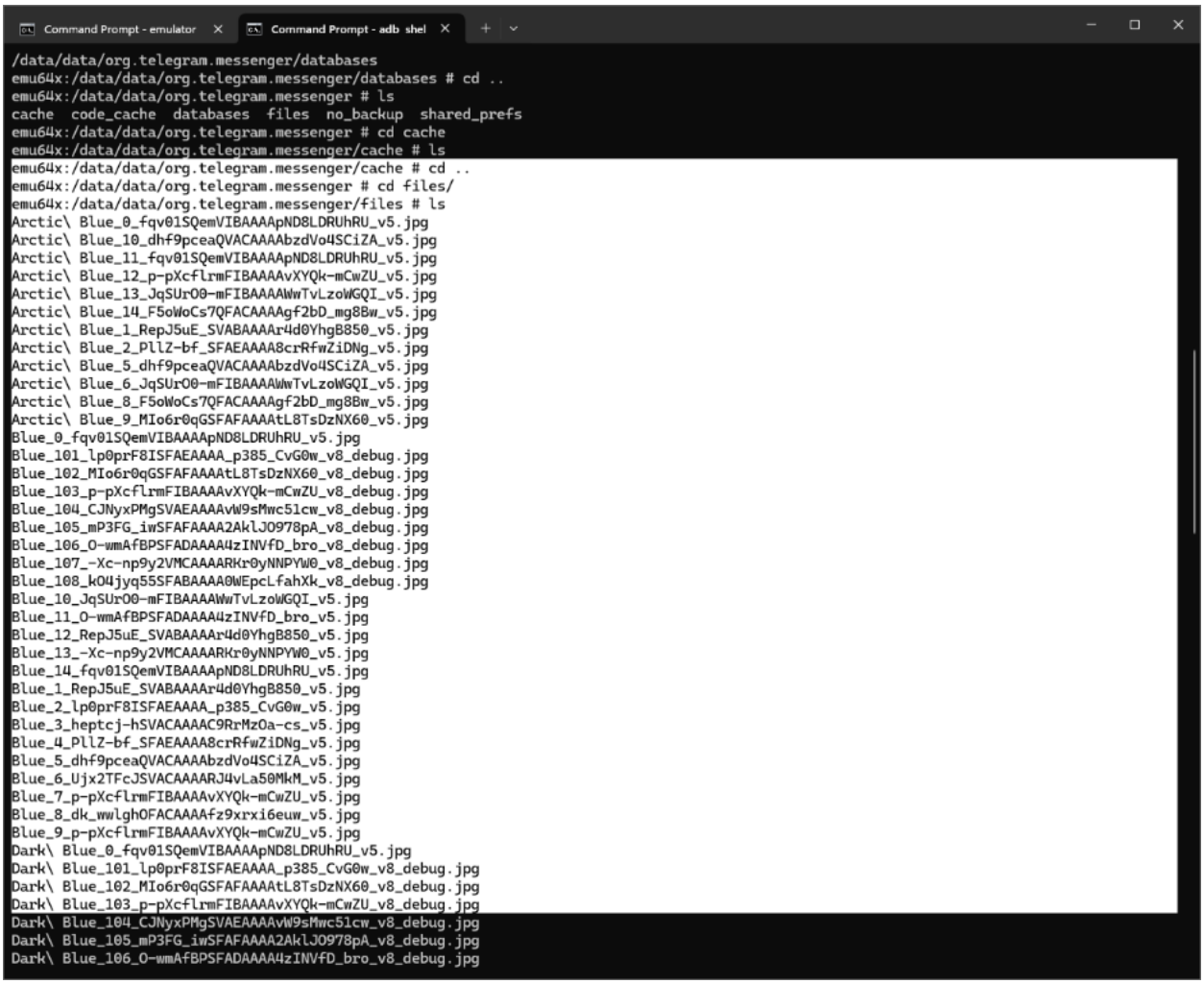
- Downloaded and installed the Telegram APK (version 11.7.4, Android 6.0+, from APKMirror – Telegram FZ-LLC).
- Logged into the Telegram app using the verification code .



Data Exploration

- Located the exact path to Telegram's databases and package directories.
- Identified event and events-journal files.
- Explored the files directory, which mainly contained JPG image files.

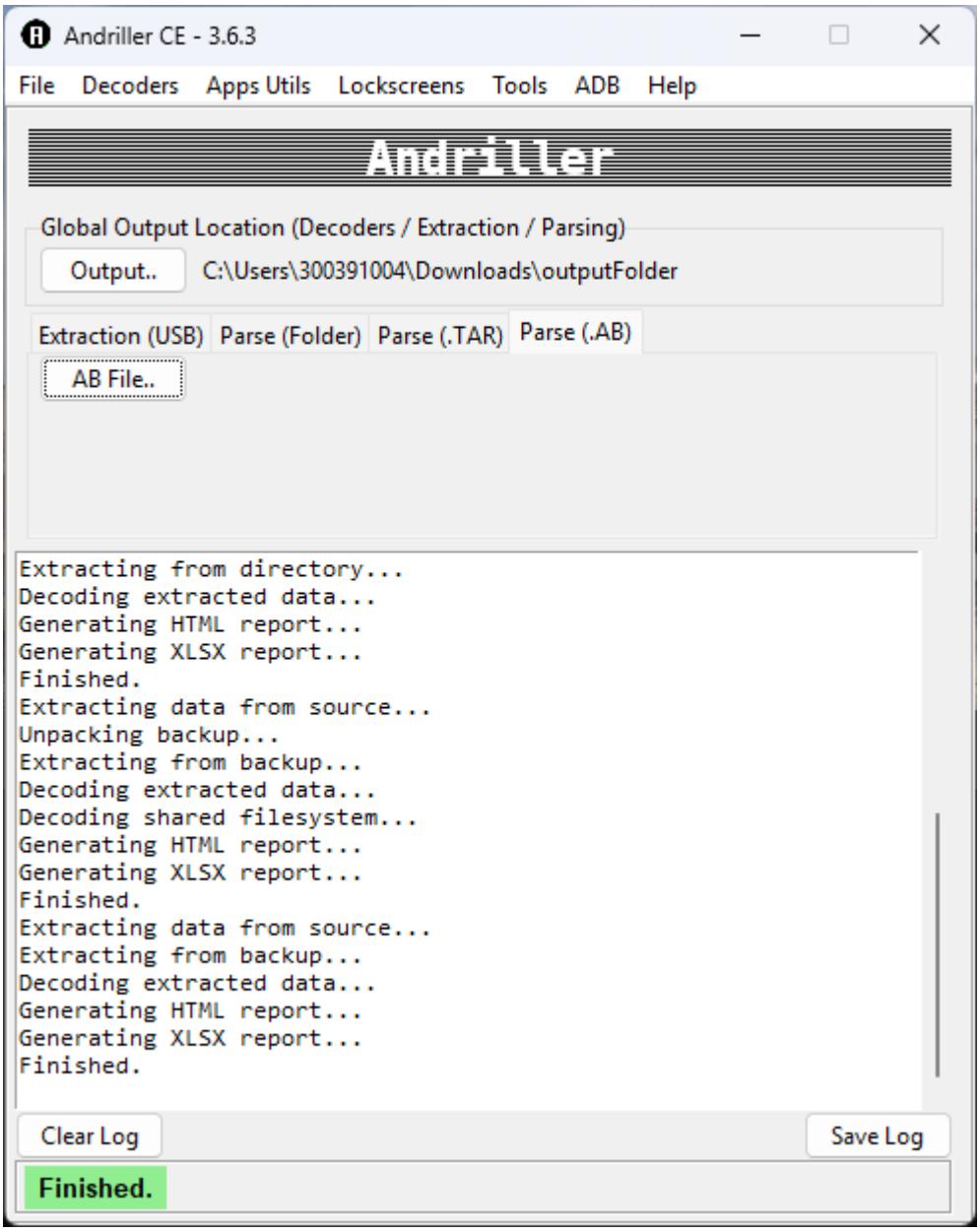



Command Prompt - emulator
Command Prompt - adb shel
/data/data/org.telegram.messenger/databases
emu64x:/data/data/org.telegram.messenger/databases # cd ..
emu64x:/data/data/org.telegram.messenger # ls
cache code_cache databases files no_backup shared_prefs
emu64x:/data/data/org.telegram.messenger # cd cache
emu64x:/data/data/org.telegram.messenger/cache # ls
emu64x:/data/data/org.telegram.messenger/cache # cd ..
emu64x:/data/data/org.telegram.messenger # cd files/
emu64x:/data/data/org.telegram.messenger/files # ls
Arctic\ Blue_0_fqv01SQemVIBAAAApND8LDRUhRU_v5.jpg
Arctic\ Blue_10_dhf9pceaQVACAAAAbzdVo4SCiZA_v5.jpg
Arctic\ Blue_11_fqv01SQemVIBAAAApND8LDRUhRU_v5.jpg
Arctic\ Blue_12_p-pXcflrmFIBAAAAvXYQk-mCwZU_v5.jpg
Arctic\ Blue_13_JqSUrO0-mFIBAAAAWwTvLzoWGQI_v5.jpg
Arctic\ Blue_14_F5oWoCs7QFACAAAAgf2bD_mg8Bw_v5.jpg
Arctic\ Blue_1_RepJ5uE_SVABAAAAr4d0YhgB850_v5.jpg
Arctic\ Blue_2_PllZ-bf_SFAEAAAA8crRfwZiDNg_v5.jpg
Arctic\ Blue_5_dhf9pceaQVACAAAAbzdVo4SCiZA_v5.jpg
Arctic\ Blue_6_JqSUrO0-mFIBAAAAWwTvLzoWGQI_v5.jpg
Arctic\ Blue_8_F5oWoCs7QFACAAAAgf2bD_mg8Bw_v5.jpg
Arctic\ Blue_9_MIo6r0qGSFAFAAAAtL8TsDzNX60_v5.jpg
Blue_0_fqv01SQemVIBAAAApND8LDRUhRU_v5.jpg
Blue_101_lp0prF8ISFAEAAAA_p385_CvG0w_v8_debug.jpg
Blue_102_MIo6r0qGSFAFAAAAtL8TsDzNX60_v8_debug.jpg
Blue_103_p-pXcflrmFIBAAAAvXYQk-mCwZU_v8_debug.jpg
Blue_104_CJNyxPMgSVAEAAAAvW9sMwc51cw_v8_debug.jpg
Blue_105_mP3FG_iwSFAFAAAA2AklJO978pA_v8_debug.jpg
Blue_106_O-wmAfBPSFADAAAA4zINVfD_bro_v8_debug.jpg
Blue_107_-Xc-np9y2VMCAAAARKr0yNNPYW0_v8_debug.jpg
Blue_108_kO4jyq55SFABAAAA0WEpcLfahXk_v8_debug.jpg
Blue_10_JqSUrO0-mFIBAAAAWwTvLzoWGQI_v5.jpg
Blue_11_O-wmAfBPSFADAAAA4zINVfD_bro_v5.jpg
Blue_12_RepJ5uE_SVABAAAAr4d0YhgB850_v5.jpg
Blue_13_-Xc-np9y2VMCAAAARKr0yNNPYW0_v5.jpg
Blue_14_fqv01SQemVIBAAAApND8LDRUhRU_v5.jpg
Blue_1_RepJ5uE_SVABAAAAr4d0YhgB850_v5.jpg
Blue_2_lp0prF8ISFAEAAAA_p385_CvG0w_v5.jpg
Blue_3_heptcj-hSVACAAAAC9RrMzOa-cs_v5.jpg
Blue_4_PllZ-bf_SFAEAAAA8crRfwZiDNg_v5.jpg
Blue_5_dhf9pceaQVACAAAAbzdVo4SCiZA_v5.jpg
Blue_6_Ujx2TFcJSVACAAAARJ4vLa50MkM_v5.jpg
Blue_7_p-pXcflrmFIBAAAAvXYQk-mCwZU_v5.jpg
Blue_8_dk_wwlghOFACAAAAfz9xrxi6euw_v5.jpg
Blue_9_p-pXcflrmFIBAAAAvXYQk-mCwZU_v5.jpg
Dark\ Blue_0_fqv01SQemVIBAAAApND8LDRUhRU_v5.jpg
Dark\ Blue_101_lp0prF8ISFAEAAAA_p385_CvG0w_v8_debug.jpg
Dark\ Blue_102_MIo6r0qGSFAFAAAAtL8TsDzNX60_v8_debug.jpg
Dark\ Blue_103_p-pXcflrmFIBAAAAvXYQk-mCwZU_v8_debug.jpg
Dark\ Blue_104_CJNyxPMgSVAEAAAAvW9sMwc51cw_v8_debug.jpg
Dark\ Blue_105_mP3FG_iwSFAFAAAA2AklJO978pA_v8_debug.jpg
Dark\ Blue_106_O-wmAfBPSFADAAAA4zINVfD_bro_v8_debug.jpg

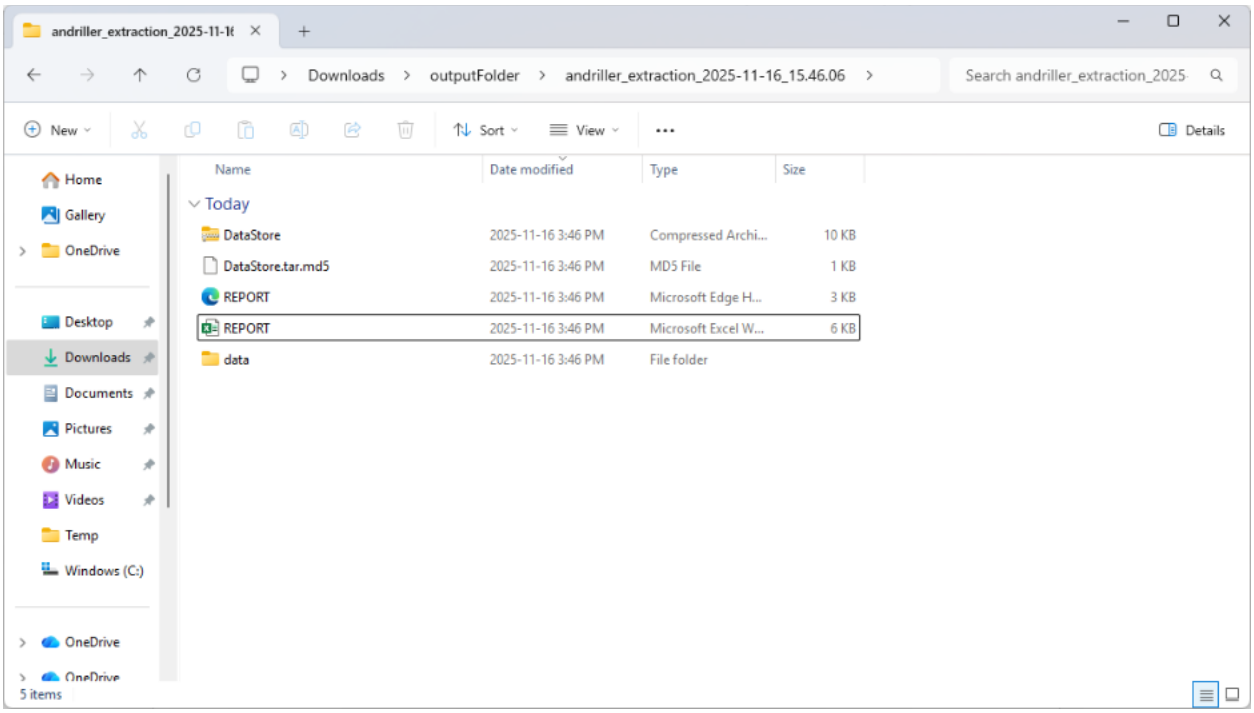
Andriller

- Installed Andriller and performed a logical backup of the device.

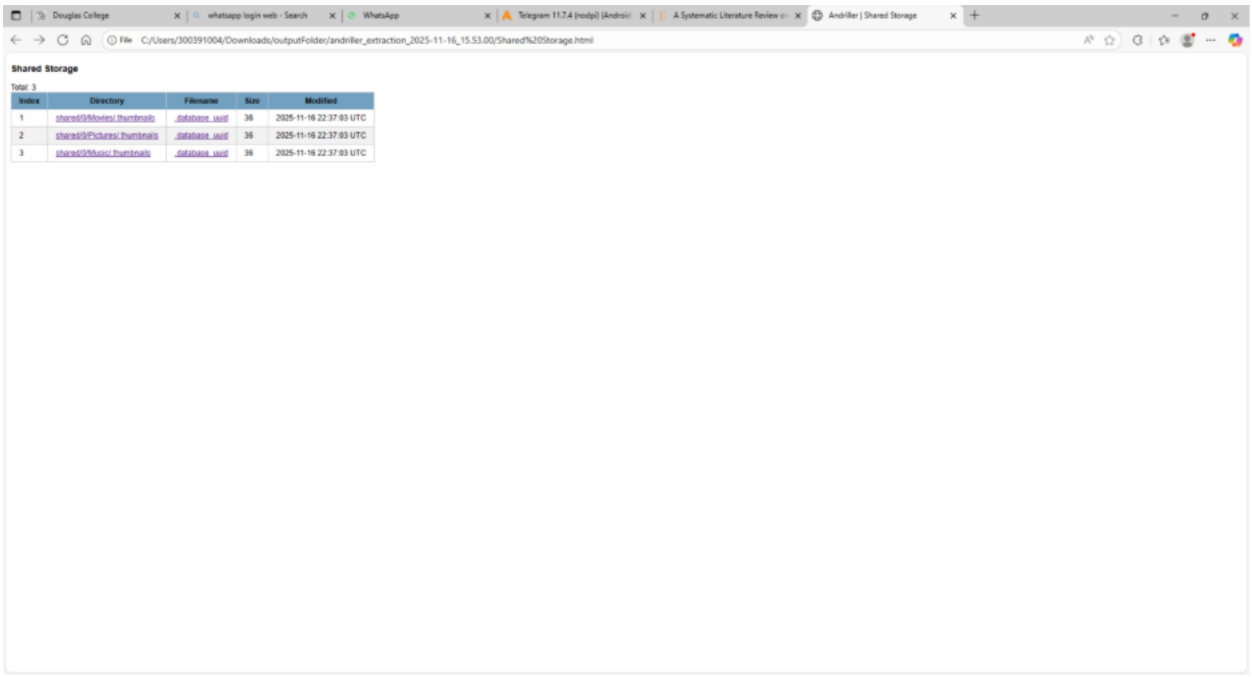
```
Command Prompt - python

Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\300391004>python -m andriller
INFO:andriller.gui.core:Started: Andriller CE 3.6.3
INFO:andriller.gui.core:Time settings/format: Y-m-d H:M:S Z
INFO:andriller.gui.core:Detected/PC time: 2025-11-16 15:45:21
INFO:andriller.gui.core:Universal time:   2025-11-16 23:45:21 UTC
INFO:andriller.gui.core:Time in reports:  2025-11-16 23:45:21 UTC <--
INFO:andriller.gui.core:Extracting from directory...
INFO:andriller.gui.core:Decoding extracted data...
INFO:andriller.gui.core:Generating HTML report...
INFO:andriller.gui.core:Generating XLSX report...
INFO:andriller.gui.core:Finished.
INFO:andriller.gui.core:Extracting from directory...
INFO:andriller.gui.core:Decoding extracted data...
INFO:andriller.gui.core:Generating HTML report...
INFO:andriller.gui.core:Generating XLSX report...
INFO:andriller.gui.core:Finished.
INFO:andriller.gui.core:Extracting from directory...
INFO:andriller.gui.core:Decoding extracted data...
INFO:andriller.gui.core:Generating HTML report...
INFO:andriller.gui.core:Generating XLSX report...
INFO:andriller.gui.core:Finished.
INFO:andriller.gui.core:Extracting from directory...
INFO:andriller.gui.core:Decoding extracted data...
INFO:andriller.gui.core:Generating HTML report...
INFO:andriller.gui.core:Generating XLSX report...
INFO:andriller.gui.core:Finished.
INFO:andriller.gui.core:Extracting data from source...
```



- Loaded the backup into Andriller and generated the reports (3 shared reports).



- The Shared Storage report mainly contained system thumbnail metadata (e.g., _database_uuid entries under .thumbnails folders), with no meaningful user media.



- This indicates that on a modern Android version, a non-root logical backup mostly provides access to shared storage and does **not** include Telegram's internal, encrypted chat databases.

Additional Analysis

- Extracted and inspected the .tar file from the backup but found no useful evidence.

Physical Backup Attempt

- Attempted a physical backup using ncat and toybox to obtain a .dd image.
- Discovered that Andriller does not support direct analysis of .dd files.

Uber App Issue

- Tried to install the Uber app, but it is built for ARM architecture.
- The app failed to install on the x86 emulator, confirming the architecture incompatibility.