Contents

COMPREHENSIVE ANALYSIS OF ANDROID MOBILE APPLICATIONS: TINDER, IMO, REDDIT AND TIM HORTONS

Course: CSIS 4440_003

Team Members:

- **Bright Ekeator - 300318200 - Team Lead**
- **Ifeoluwa Aribo - 300389564**

Video Presentation Link 1: https://youtu.be/Q_rLYO5uJBk

Video Presentation Link 2: https://collegedouglas-my.sharepoint.com/:v:/g/personal/ariboi_student_douglascollege_ca/EWuvyGVOzHdNhFut5NmKGjYBiEYgyq4dhwfqB_0VOOY3Hg?e=fblhOj

ABSTRACT

This project conducts forensic and security analysis of popular Android applications using specialized tools. Andriller extracted communication artifacts from IMO and Tinder, while MobSF and AndroBugs identified critical vulnerabilities in Reddit and Tim Hortons, including insecure cryptography and data exposure. The research demonstrates practical mobile forensic techniques for investigative and security assessment scenarios.

1.0 BACKGROUND RESEARCH

1.1 Related Tools or Research

Mobile device forensics has emerged as a fundamental element of contemporary digital investigations because of the widespread use of technologies especially smartphones and messaging applications in both legal and illegal activities. Numerous forensic tools are available to assist with Android extraction and analysis, including:

- **Andriller (Android Forensic Toolkit)**: A well-known forensic suite capable of ADB extraction, decoding of common Android apps, data carving, password pattern decoding, and automated reporting. It is widely referenced in academic and professional mobile forensics research due to its ability to process Android backups (.ab), TAR archives, and emulator extractions.
- **Autopsy / Sleuth Kit**: Offers broader digital forensics capabilities, including mounting dd images, file system analysis, timeline correlation, and keyword searching.
- **MobSF** : Mobile security and forensic analysis platforms useful for static/dynamic analysis of APKs and file systems.

  MobSF provides comprehensive analysis including:

  - Manifest file security analysis
  - Certificate and signing validation
  - Code analysis for security anti-patterns
  - Hardcoded secrets detection
  - Tracker and permission analysis
  - Binary analysis capabilities

- **Androbug Framework**: A mobile application vulnerability and analysis tool used to inspect APK internals, permissions, and security posture.

  AndroBugs Framework specializes in deep static analysis, offering:

  - Component exposure analysis

- SSL/TLS implementation verification
- WebView security assessment
- Cryptographic implementation review
- File operation security analysis

Academic studies confirm that while modern communication apps often employ encryption, they invariably leave forensic traces on devices, including metadata, cached files, and authentication tokens. To uncover this evidence, a multi-faceted approach is essential. This includes **forensic extraction tools like Andriller**, which recover application data and artifacts from devices, and **security analysis techniques**. The latter employs Static Application Security Testing (SAST) with tools like MobSF and AndroBugs to scan APK files for code vulnerabilities based on OWASP guidelines. This project builds on this integrated methodology, applying both forensic extraction with Andriller and security analysis to applications like IMO, Tinder, Reddit, and Tim Hortons, demonstrating a comprehensive approach to uncovering digital evidence and security flaws..

1.2 Purpose of the Tool / Forensic Examination

The purpose is to examine what digital data can be recovered, analyzed, and validated from IMO, Tinder, Reddit, and Tim Hortons in a controlled Android environment using Andriller. The goal is to extract and review application data and evaluate the security posture of each platform through a complete forensic workflow.

The analysis focuses on data directories, internal storage files, databases, logs, shared preferences, and cached media. You collect message metadata, contact links, timestamps, match activity traces, login records, and communication remnants. You identify user accounts, device identifiers, authentication tokens, and app configuration files across all applications.

You evaluate data collection behaviour, third party integrations, and privacy risks. You check for misconfigurations, insecure storage, unencrypted sensitive data, weak API handling, and risks linked to social interaction, communication, and mobile payment features. You assess each application against OWASP Mobile Top 10, GDPR, CCPA, and PCI DSS where relevant.

You review timeline artefacts and metadata that persist even when message content is encrypted. This includes traces of chats, calls, payments, matches, media exchanges, login activity, and preference modifications.

Using Andriller supports automated extraction, decoding, HTML reporting, and structured database review. You apply practical forensic methods used in mobile incident response, activity correlation, evidence preservation, and security auditing.

The combined objective is to deliver a unified forensic and security evaluation of IMO, Tinder, Reddit, and Tim Hortons. This covers communication activity, social behaviour, mobile commerce activity, and application level security risks. The goal is to identify shared weaknesses, app specific risks, and broader patterns in mobile security and privacy exposure.

1.3 Forensic Scenarios Where the Tool/App Will Be Useful

Scenario 1: Suspected Online Grooming / Harassment Case (IMO)

Investigators might obtain a mobile device from a minor or victim who has communicated via chat or video with an unidentified person on IMO, a widely used app for instant messaging and video calls. Even if the suspect removes messages, the device might still hold:

- Residual chat database entries
- Contact associations
- Timestamps showing when conversations occurred
- Cached media files or call logs
- Metadata revealing the suspect's IMO ID

With Andriller, an investigator can retrieve the IMO data directory, maintain evidence in a forensically valid way, and reconstruct communication trends that either support or challenge allegations

Scenario 2: Infidelity, Stalking, or Identity Fraud Case (Tinder)

Swipes, matches, profile changes, login timestamps, and message exchanges are all recorded by the match-based social networking app Tinder. A forensic examiner may need to gather information in a civil or criminal inquiry (such as cyberstalking, impersonation, extortion, or harassment):

- Match history and metadata
- Profile information
- User IDs and tokens

- Message timestamps (even if the content is encrypted or partially deleted)
- Geolocation hints from logs or cached files

Andriller enables systematic extraction and decoding of Tinder artefacts stored within SQLite databases and app-specific directories, helping investigators correlate online behaviour with real-world events.

Scenario 3: Mobile Payment Fraud Investigation

Context: A financial institution investigates fraudulent transactions originating from mobile payment applications. Multiple customers report unauthorized transactions through the Tim Hortons mobile app.

Forensic Application:

- Static analysis reveals critical SSL certificate validation bypass vulnerabilities allowing Man-in-the-Middle attacks
- Shared preferences examination exposes plaintext authentication tokens and AWS Cognito credentials
- The insecure cryptographic implementation (CBC mode without authentication) identified in our analysis demonstrates potential for credential decryption
- Dynamic analysis captures API communications showing payment processing flows and authentication mechanisms

Investigative Value:

- Authentication token extraction and validation
- Payment transaction timeline reconstruction
- Network traffic analysis for interception evidence
- Identification of vulnerable communication channels exploited by attackers
- Recovery of loyalty card information and user PII for victim identification

Scenario 4: Malware Analysis and Attribution

Context: A cybersecurity firm analyzes a sophisticated mobile malware that mimics legitimate application behavior to evade detection.

Forensic Application:

- Static analysis techniques demonstrated in this project enable identification of malicious code patterns hidden within seemingly legitimate applications

- Component exposure analysis reveals attack surfaces exploitable by the malware
- Dynamic analysis captures malware communication with command-and-control servers
- Comparison with baseline application behavior identifies anomalous activities

Investigative Value:

- Malware capability assessment through code analysis
- Command-and-control infrastructure identification
- Attribution through coding patterns and infrastructure analysis
- Impact assessment on compromised devices

2.0 FORENSIC PLAN

2.1 How the Tools Work

Andriller

Andriller is an Android forensic toolkit that automates the extraction, decoding, and reporting of digital evidence from Android devices. It works by leveraging:

- **ADB (Android Debug Bridge)** to communicate with an Android device or emulator
- **Backup extraction** (AB → TAR → file system reconstruction)
- **App-specific decoders** for popular apps
- **SQLite parsing** to read databases
- **Automated HTML report generation** for examiners

Andriller operates in three primary phases:

1. **Device Interaction:**
   Connects to the Android device/emulator through ADB, enabling file pulls, application data access, and querying of system properties.
2. **Extraction Phase:**
   Creates an Android backup (.ab) or extracts directories directly using TAR/AirGap

extraction. The tool parses the resulting archive to recover application folders such as:

   - /data/data/com.imo.android.imoim/
   - /data/data/com.tinder/

3. **Decoding & Reporting:**
   Andriller examines extracted databases (e.g., messages.db, cache.db, shared_prefs) and converts them into readable formats. It generates:
   - HTML tables
   - CSV files
   - Parsed JSON artefacts
   - Timeline summaries

This automated workflow reduces manual parsing time and ensures consistency across examinations.

MobSF (Mobile Security Framework) Architecture:

MobSF is an open-source, automated mobile application security testing framework that performs static, dynamic, and malware analysis. The framework architecture consists of:

1. **APK Decompilation Engine:**
   - Utilizes apktool and dex2jar for reverse engineering
   - Extracts AndroidManifest.xml, resources, and DEX (Dalvik Executable) files
   - Converts DEX bytecode to Java source code using multiple decompilers (Jadx, CFR)
   - Handles obfuscated code through pattern recognition algorithms
2. **Static Analysis Module:**
   - **Manifest Analysis:** Parses AndroidManifest.xml to identify component declarations, permissions, exported components, and security configurations
   - **Code Analysis:** Implements regex-based pattern matching and Abstract Syntax Tree (AST) analysis to detect security anti-patterns

    - **Binary Analysis:** Examines native libraries (.so files) for potential vulnerabilities
    - **Resource Analysis:** Scans strings.xml, assets, and resources for hardcoded secrets

3. **Security Scanning Engine:**
    - Implements CWE and OWASP Mobile Top 10 detection rules
    - Uses signature-based detection for known vulnerabilities
    - Performs cryptographic algorithm analysis
    - Identifies insecure network configurations
4. **Dynamic Analysis Module:**
    - Instruments applications using Frida framework
    - Captures network traffic through mitmproxy
    - Monitors file system operations
    - Logs API calls and method invocations
    - Tracks runtime permission requests

AndroBugs Framework Architecture:

AndroBugs is a specialized Android vulnerability scanner focusing on comprehensive static analysis with detailed reporting.

1. **Analysis Engine:**
    - Python-based modular architecture
    - Multi-pass analysis for deep inspection
    - Custom vulnerability detection patterns
    - Component interaction analysis
2. **Vulnerability Detection Modules:**
    - **SSL/TLS Security:** Detects custom TrustManagers, hostname verifiers, and certificate validation bypasses

- **Component Security:** Analyzes exported components, intent filters, and permission protection
- **WebView Security:** Identifies JavaScript interface risks, file access configurations, and debugging settings
- **Cryptographic Analysis:** Detects weak algorithms, insecure key storage, and improper cryptographic implementations
- **Code Quality:** Identifies insecure coding patterns, file operations, and data handling issues

3. **Reporting System:**
    - Severity-based classification (CRITICAL, WARNING, NOTICE, INFO)
    - CVSS scoring for critical findings
    - Remediation recommendations with code examples
    - Detailed technical documentation of each finding

Dynamic Analysis Methodology:

Dynamic analysis involves executing the application in a controlled environment while monitoring its behavior:

1. **Environment Setup:**
    - Android emulator or rooted physical device
    - System call monitoring tools (strace, ltrace)
2. **Instrumentation:**
    - Frida scripts inject monitoring code into running applications
    - Xposed Framework modifies system behavior
    - Hooking frameworks intercept method calls
3. **Behavior Monitoring:**
    - Network traffic capture and SSL/TLS decryption
    - File system access logging
    - Database operation tracking

    - API call interception
    - Runtime permission monitoring

2.2 How the Examination Will Be Conducted

The forensic examination for this project follows a **structured, repeatable process** to ensure evidence integrity and academic reproducibility.

Step 1: Environment Setup

- Use **Android Studio Emulator** to create a test Android device.
- Install IMO and Tinder APKs.
- Confirming device connection using "adb devices" command to confirm the right device is connected to the Andriller.

Step 2: Evidence Preservation

Before analysis, a clean extraction of the device state is performed using Andriller's ADB extraction features.

- Generate an Android backup (.ab file) OR
- Pull the /data/data/ directories using Andriller's file extraction module.

This ensures forensic integrity by capturing the device in its current state without modifying inner artefacts.

Step 3: Data Extraction

Using Andriller:

- Load the .ab or TAR archive into the tool
- Allow Andriller to extract and reconstruct the file system
- Observe generated folders such as:
    - com.imo.android.imoim/ (chat data, logs, prefs, media)
    - com.tinder/ (user metadata, matches, messages, tokens)

Step 4: Database & Artefact Analysis

Using other tools (DB browser, Dcode. etc) to Inspect the results generated by Andriller, including:

- SQLite databases (message logs, contacts, chat metadata, match information)
- Shared preference XML files (authentication tokens, user IDs, timestamps)
- Cached media (images, profile photos, video thumbnails)
- Logs and metadata
- Decoded HTML reports

This phase includes aligning timestamps, recognizing communication trends, and examining extracted artefacts for forensic significance

Step 5: Documentation & Reporting

All findings will be:

- Compiled into a coherent forensic report
- Supported with screenshots, extracted tables, and parsed outputs

FOR TIMHORTONS AND REDDIT

The security and forensic analysis for this project follows a **structured, repeatable process** to ensure evidence integrity and academic reproducibility, divided into three core phases: Static Analysis, Dynamic Analysis, and Integrated Forensic Analysis.

Phase 1: Static Analysis (Code & Configuration Examination)

Objective: To identify vulnerabilities, misconfigurations, and insecure coding practices without executing the application.

Step 1: APK Acquisition and Environment Setup

- Acquire the target APK files (com.reddit.frontpage and digital.rbi.timhortons) from a controlled source.
- Set up the static analysis environment, including:
  - **MobSF (Mobile Security Framework)** for automated comprehensive scanning.
  - **AndroBugs Framework** for deep, signature-based vulnerability detection.
  - **jadx-gui** for manual source code review and decompilation.

Step 2: Automated Static Scanning

- Upload each APK to **MobSF** and execute a full static scan to generate a security score and identify issues like insecure cryptography, hardcoded secrets, and permission misuse.

- Run **AndroBugs Framework** via command line to perform a targeted scan for critical vulnerabilities such as SSL bypasses, exported components, and WebView misconfigurations.

Step 3: Manual Code Review & Artefact Analysis

- Use **jadx-gui** to manually inspect decompiled source code, focusing on areas flagged by automated tools.
- Analyze the AndroidManifest.xml for exported components, permission requests, and backup settings.
- Examine resource files for hardcoded API keys and sensitive configuration data.

Step 4: Documentation of Static Findings

- Compile all vulnerabilities with their respective severity levels (Critical, Warning, Info).
- Document code snippets, file paths, and specific misconfigurations for later correlation.

Phase 2: Dynamic Analysis (Runtime Behavior Examination)

Objective: To observe the application's runtime behavior, network traffic, and data handling in a controlled environment.

Step 1: Dynamic Environment Setup

- Configure an **Android Studio Emulator** (or rooted physical device) with a compatible Android version.
- Install the target APKs via ADB.
- Initiate the **MobSF Dynamic Analysis** module, which will instrument the environment with Frida and configure network traffic interception via mitmproxy.

Step 2: Interactive Application Testing

- Execute core application workflows:
  - **Reddit:** User login, browsing subreddits, viewing posts and images.
  - **Tim Hortons:** Account login, browsing the menu, simulating a loyalty points check.
- Monitor and log all runtime behavior, including system calls, file operations, and database interactions.

Step 3: Network Traffic Interception

- Capture and analyze all HTTP/HTTPS traffic using mitmproxy to identify:

    - API endpoints and data exchanged.
    - Communication with third-party tracking and analytics services.
    - Potential transmission of sensitive data in cleartext.

Step 4: Data Storage Forensics

- Use ADB commands to pull the application's data directory (/data/data/<package_name>).
- Forensically examine extracted data using **DB Browser for SQLite** and a text editor, focusing on:
    - SQLite databases for user data, cached content, and session information.
    - SharedPreferences XML files for authentication tokens, user identifiers, and configuration.
    - Cache directories for temporary files and images.

Phase 3: Integrated Forensic Analysis & Correlation

Objective: To synthesize findings from all phases, validate vulnerabilities, and assess the real-world forensic impact.

Step 1: Evidence Correlation and Validation

- Cross-reference findings from Static and Dynamic analysis to eliminate false positives/negatives.
- Confirm the exploitability of static code vulnerabilities (e.g., SSL bypass) with evidence from dynamic traffic analysis.
- Validate that theoretical risks (e.g., insecure data storage) are realized with actual extracted artefacts (e.g., plaintext tokens in SharedPreferences).

Step 2: Forensic Artefact Analysis

- Analyze the extracted application data to build a forensic profile of user activity.
- Reconstruct user timelines and behaviors based on database timestamps, logs, and cached media.
- Identify forensically significant artefacts such as loyalty card details, authentication tokens, and personal identifiers.

Step 3: Documentation & Reporting

- Compile all findings into a coherent forensic and security report.
- Support conclusions with:

    - **Screenshots** of tool outputs, decompiled code, and application behavior.
    - **Extracted tables and data** from databases and configuration files.
    - **Parsed outputs** from automated tools, correlated with manual findings.
- Provide a final risk assessment and remediation roadmap for each application.

2.3 Underlying Implementation / Architecture Needed

To successfully demonstrate the forensic analysis, the following architecture is required:
1. Technical Setup

- **Windows Host Machine** running: Android Studio, ADB tools, IMO/Tinder AND TimHortons/Reddit APKs from an official source(Google Playstore).

2. Connectivity Architecture

The Android emulator is connected to the forensic workstation (Andriller) via ADB bridge
Primary Tools:

- MobSF v4.4.3 (latest stable)

- Andriller

- AndroBugs Framework v1.0.0

- Frida v16.x

Supporting Tools:

- jadx-gui v1.4.7 (decompilation)

- apktool v2.9.0 (resource extraction)

- DB Browser for SQLite v3.12

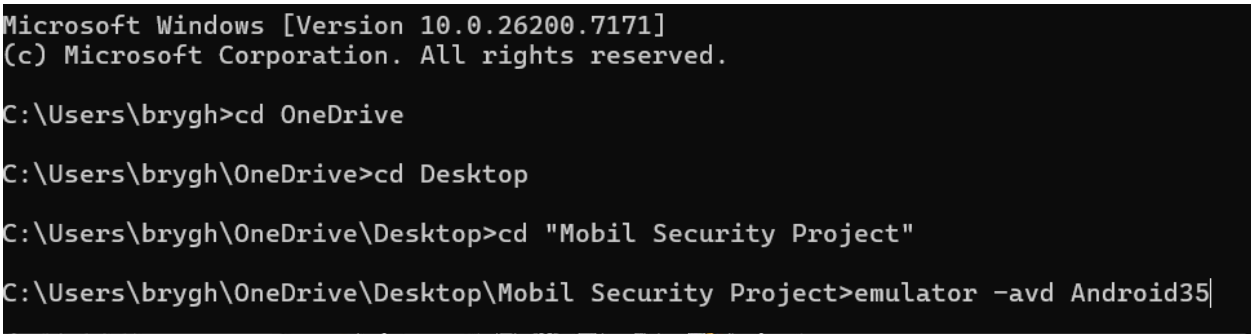
- ADB (Android Debug Bridge)

3.0 DETAILED ANALYSIS

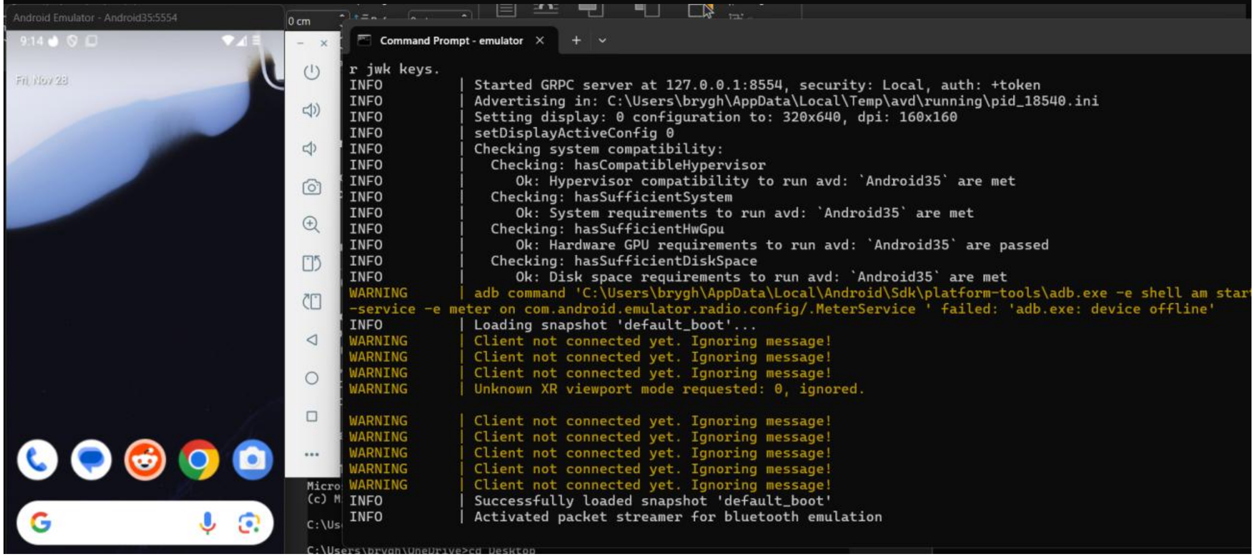
This part outlines the complete practical forensic procedure conducted on the IMO and Tinder apps with Andriller, accompanied by screenshots, command results, file-system proof, and insights from both successful and failed extraction efforts. This assessment mimics an actual forensic process and records the difficulties faced during extraction and decoding

3.1 Hands-On Demonstration: Connecting and Preparing the Android Emulator

Step 1: Launching the Android Emulator



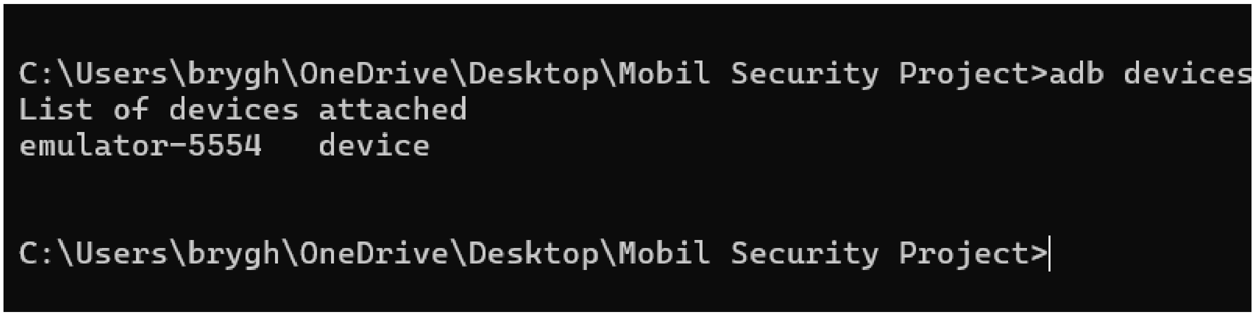
```
Microsoft Windows [Version 10.0.26200.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\brygh>cd OneDrive

C:\Users\brygh\OneDrive>cd Desktop

C:\Users\brygh\OneDrive\Desktop>cd "Mobil Security Project"

C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>emulator -avd Android35
```



```
r jwk keys.
INFO         | Started GRPC server at 127.0.0.1:8554, security: Local, auth: +token
INFO         | Advertising in: C:\Users\brygh\AppData\Local\Temp\avd\running\pid_18540.ini
INFO         | Setting display: 0 configuration to: 320x640, dpi: 160x160
INFO         | setDisplayActiveConfig 0
INFO         | Checking system compatibility:
INFO         |   Checking: hasCompatibleHypervisor
INFO         |      Ok: Hypervisor compatibility to run avd: `Android35` are met
INFO         |   Checking: hasSufficientSystem
INFO         |      Ok: System requirements to run avd: `Android35` are met
INFO         |   Checking: hasSufficientHwGpu
INFO         |      Ok: Hardware GPU requirements to run avd: `Android35` are passed
INFO         |   Checking: hasSufficientDiskSpace
INFO         |      Ok: Disk space requirements to run avd: `Android35` are met
WARNING      | adb command 'C:\Users\brygh\AppData\Local\Android\Sdk\platform-tools\adb.exe -e shell am star
-service -e meter on com.android.emulator.radio.config/.MeterService ' failed: 'adb.exe: device offline'
INFO         | Loading snapshot 'default_boot'...
WARNING      | Client not connected yet. Ignoring message!
WARNING      | Client not connected yet. Ignoring message!
WARNING      | Client not connected yet. Ignoring message!
WARNING      | Unknown XR viewport mode requested: 0, ignored.

WARNING      | Client not connected yet. Ignoring message!
WARNING      | Client not connected yet. Ignoring message!
WARNING      | Client not connected yet. Ignoring message!
WARNING      | Client not connected yet. Ignoring message!
WARNING      | Client not connected yet. Ignoring message!
INFO         | Successfully loaded snapshot 'default_boot'
INFO         | Activated packet streamer for bluetooth emulation
```

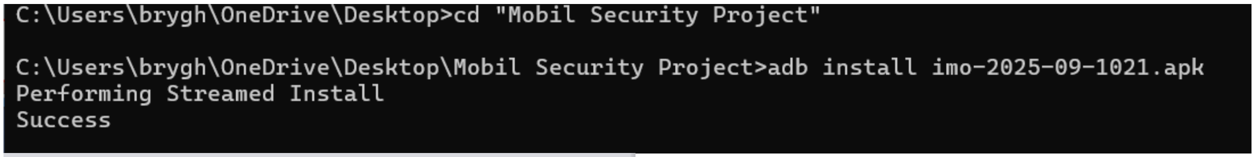
Step 2: Verifying ADB Connection

The connection between the Windows host and the emulator was validated

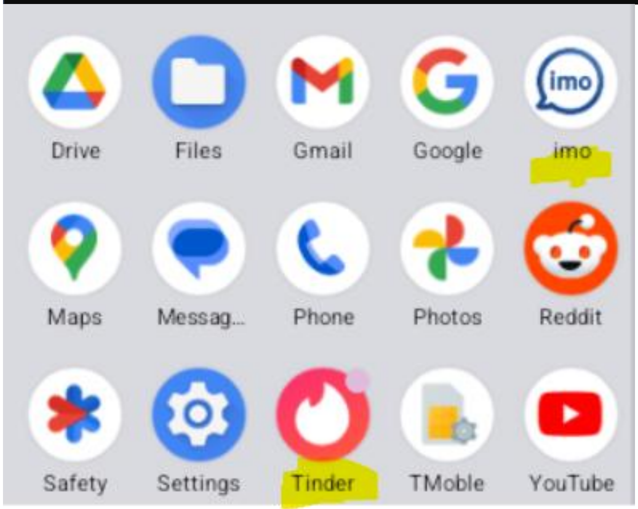


```
C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>adb devices
List of devices attached
emulator-5554   device


C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>
```

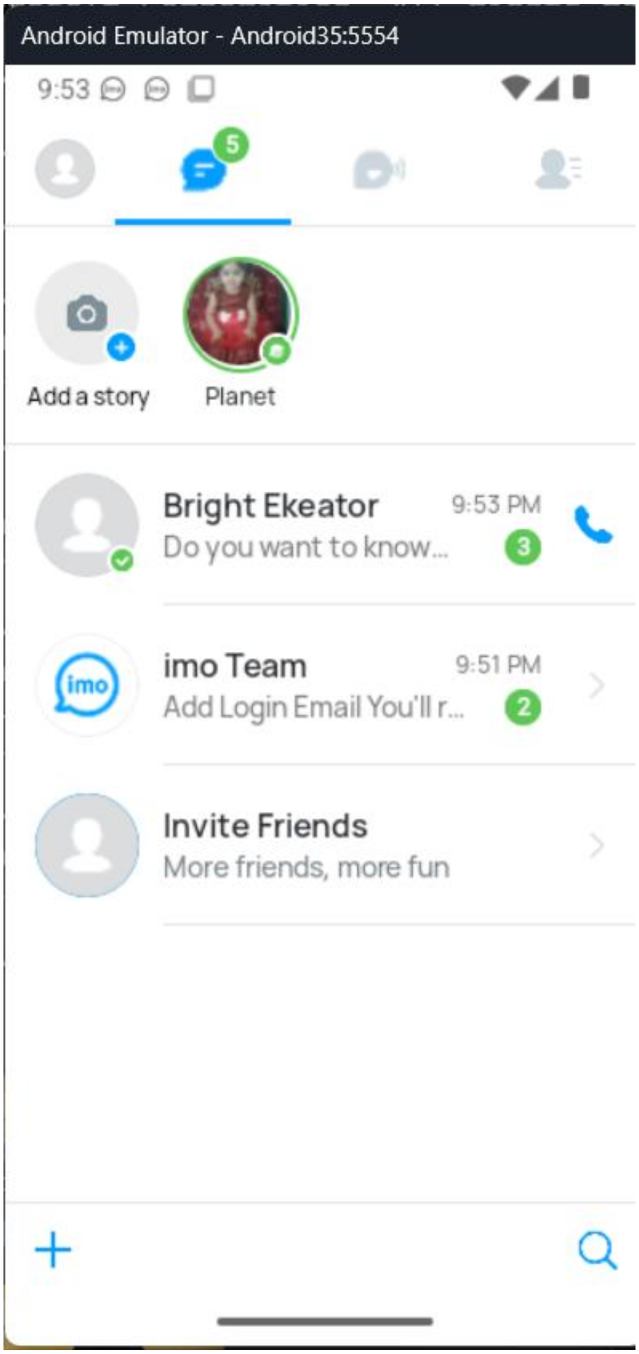
Step 3: Installing the Apps (IMO and Tinder)



```
C:\Users\brygh\OneDrive\Desktop>cd "Mobil Security Project"

C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>adb install imo-2025-09-1021.apk
Performing Streamed Install
Success
```



3.2 Populating data into the database by using the apps on the emulator

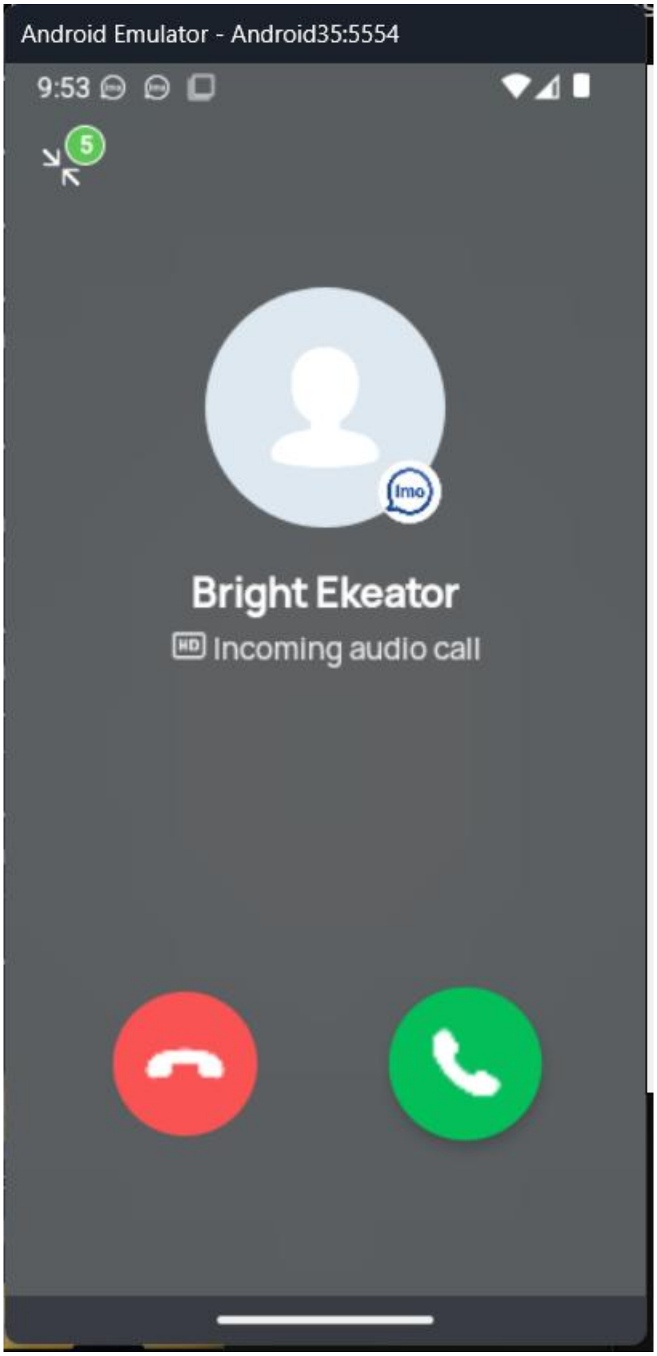


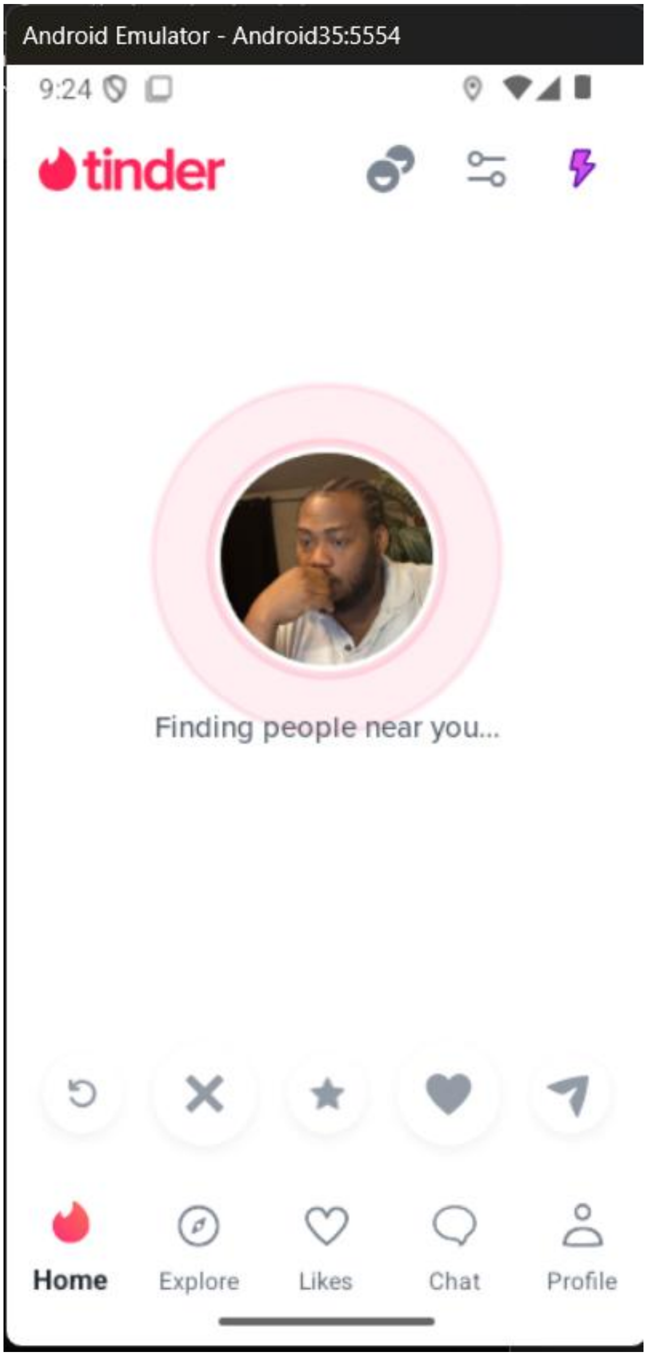
Android Emulator - Android35:5554

Bright Ekeator 9:53 PM
Do you want to know... 3

imo Team 9:51 PM
Add Login Email You'll r... 2

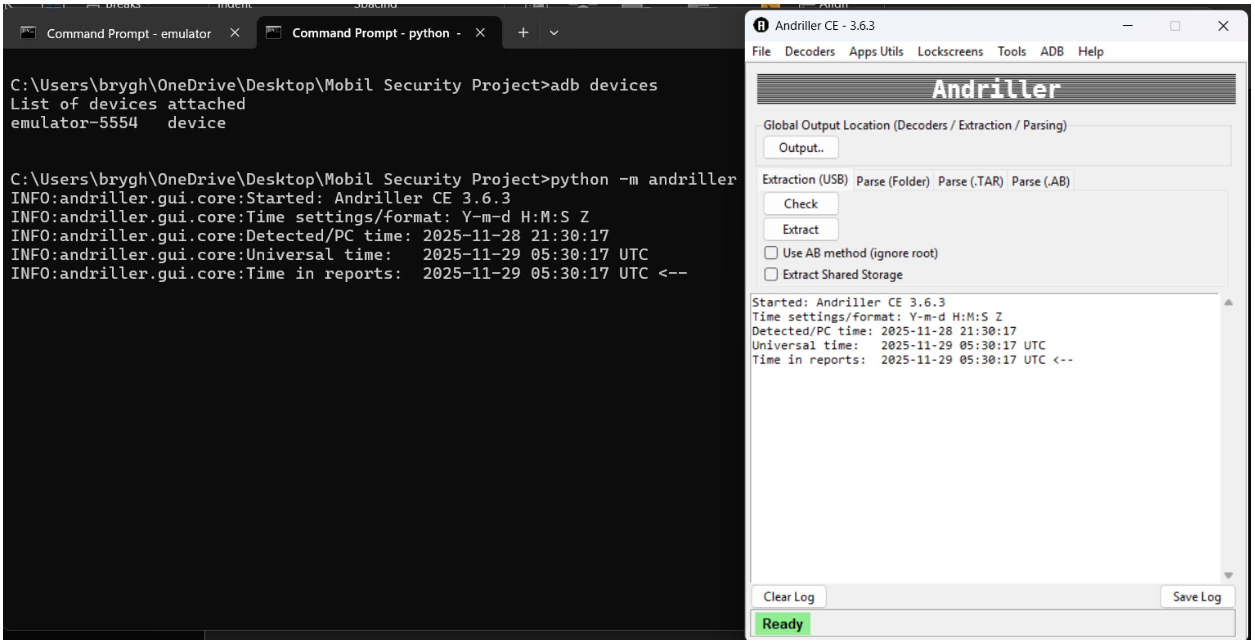
Invite Friends
More friends, more fun


Android Emulator - Android35:5554
9:53
5
imo
Bright Ekeator
HD Incoming audio call



3.3 Extraction Using Andriller

Step 1: Launching Andriller



```
C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>adb devices
List of devices attached
emulator-5554   device


C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>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: 2025-11-28 21:30:17
INFO:andriller.gui.core:Universal time:   2025-11-29 05:30:17 UTC
INFO:andriller.gui.core:Time in reports:  2025-11-29 05:30:17 UTC <--
```

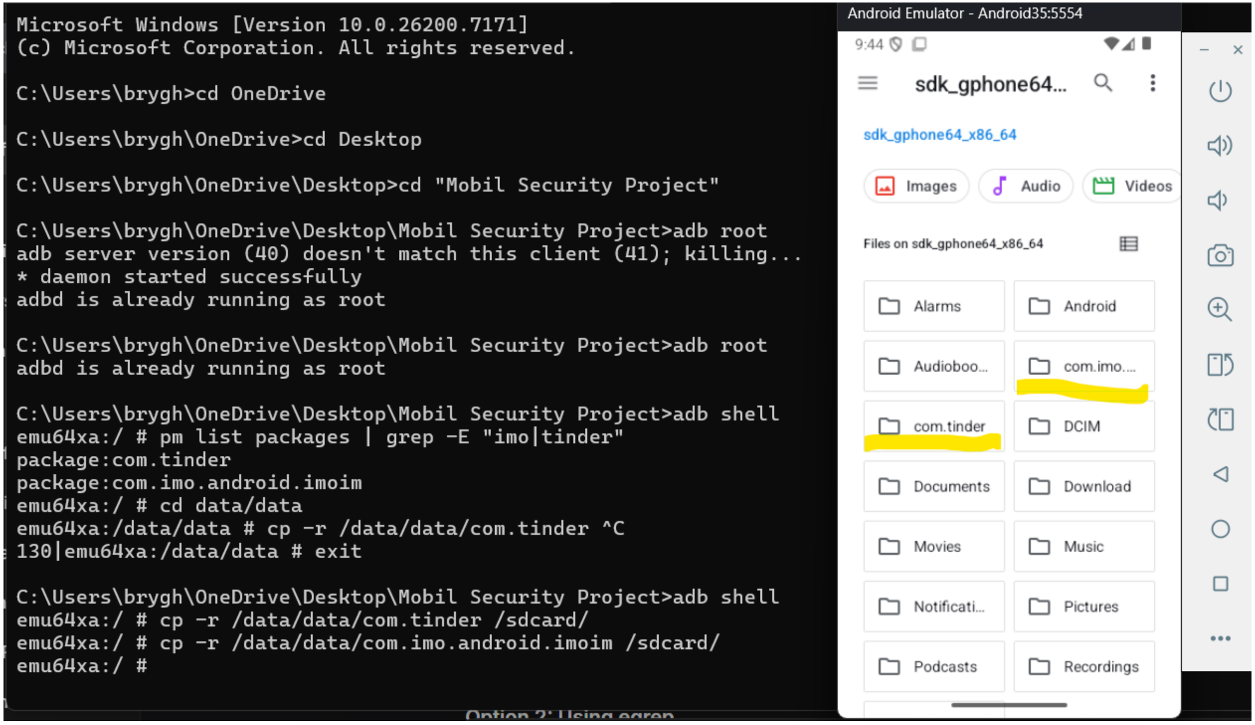
☐ Use AB method (ignore root)

☐ Extract Shared Storage

```
Started: Andriller CE 3.6.3
Time settings/format: Y-m-d H:M:S Z
Detected/PC time: 2025-11-28 21:30:17
Universal time:   2025-11-29 05:30:17 UTC
Time in reports:  2025-11-29 05:30:17 UTC <--
```

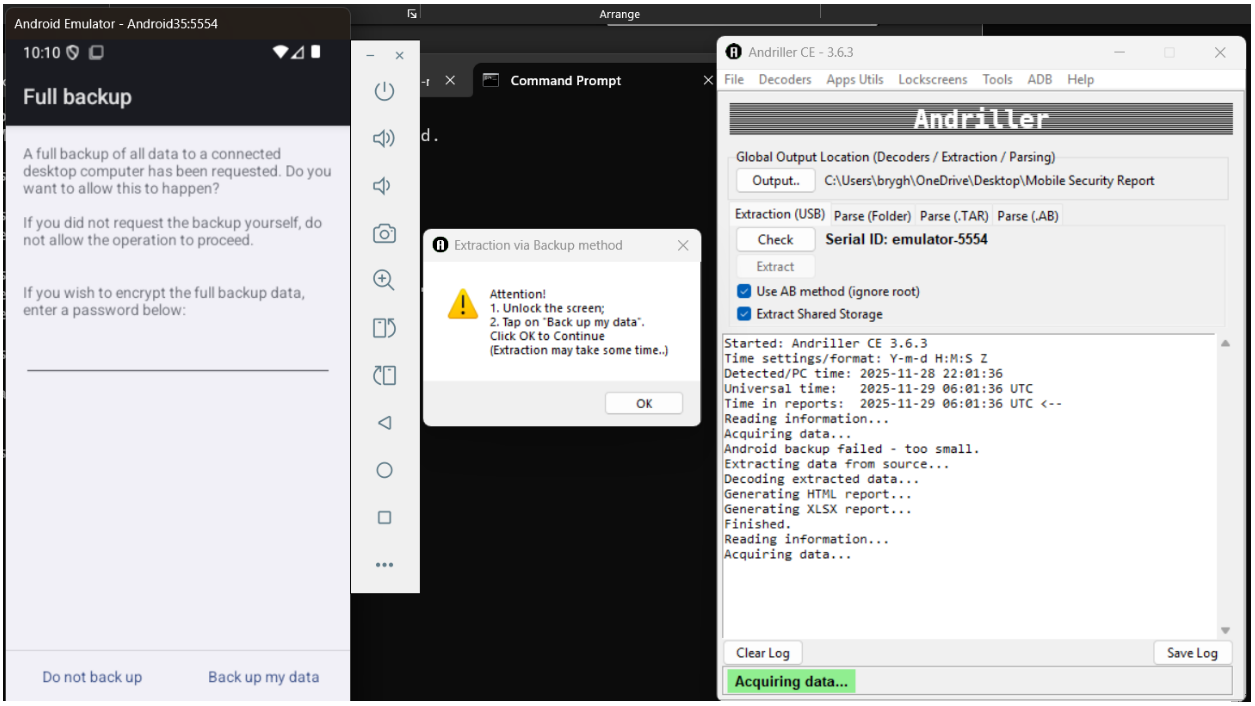
Step 2: Data Extraction.

Due to the difficulties for Andriller to extract as root, I found a way around it by copying the files from the data/data folder direct to the sdcard making it easier for extraction



```
Microsoft Windows [Version 10.0.26200.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\brygh>cd OneDrive

C:\Users\brygh\OneDrive>cd Desktop

C:\Users\brygh\OneDrive\Desktop>cd "Mobil Security Project"

C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>adb root
adb server version (40) doesn't match this client (41); killing...
* daemon started successfully
adbd is already running as root

C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>adb root
adbd is already running as root

C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>adb shell
emu64xa:/ # pm list packages | grep -E "imo|tinder"
package:com.tinder
package:com.imo.android.imoim
emu64xa:/ # cd data/data
emu64xa:/data/data # cp -r /data/data/com.tinder ^C
130|emu64xa:/data/data # exit

C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>adb shell
emu64xa:/ # cp -r /data/data/com.tinder /sdcard/
emu64xa:/ # cp -r /data/data/com.imo.android.imoim /sdcard/
emu64xa:/ #
```

Step 2: Package Extraction using Andriller



Full backup

A full backup of all data to a connected desktop computer has been requested. Do you want to allow this to happen?

If you did not request the backup yourself, do not allow the operation to proceed.

If you wish to encrypt the full backup data, enter a password below:

Extraction via Backup method

Attention!
1. Unlock the screen;
2. Tap on "Back up my data".
Click OK to Continue
(Extraction may take some time..)

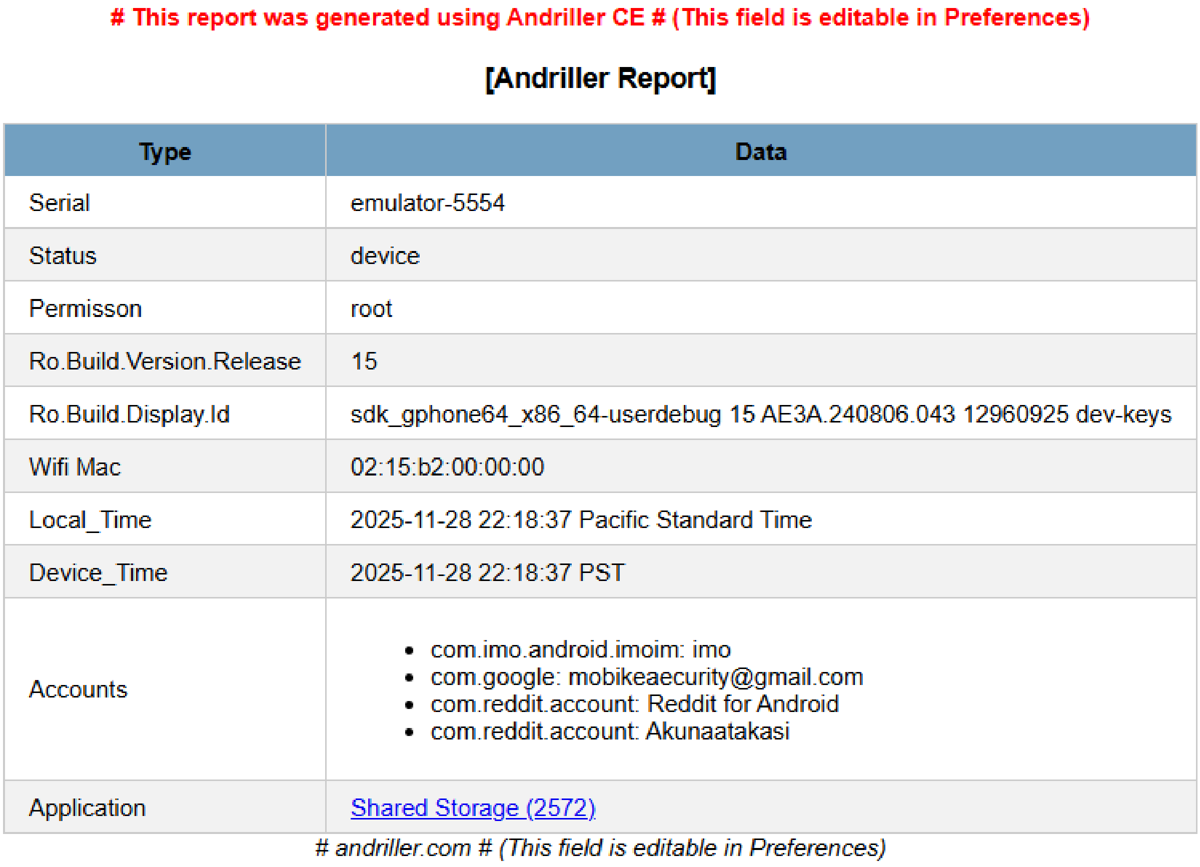
Global Output Location (Decoders / Extraction / Parsing)
C:\Users\brygh\OneDrive\Desktop\Mobile Security Report

Serial ID: emulator-5554

☑ Use AB method (ignore root)
☑ Extract Shared Storage

```
Started: Andriller CE 3.6.3
Time settings/format: Y-m-d H:M:S Z
Detected/PC time: 2025-11-28 22:01:36
Universal time:   2025-11-29 06:01:36 UTC
Time in reports:  2025-11-29 06:01:36 UTC <--
Reading information...
Acquiring data...
Android backup failed - too small.
Extracting data from source...
Decoding extracted data...
Generating HTML report...
Generating XLSX report...
Finished.
Reading information...
Acquiring data...
```

Acquiring data...

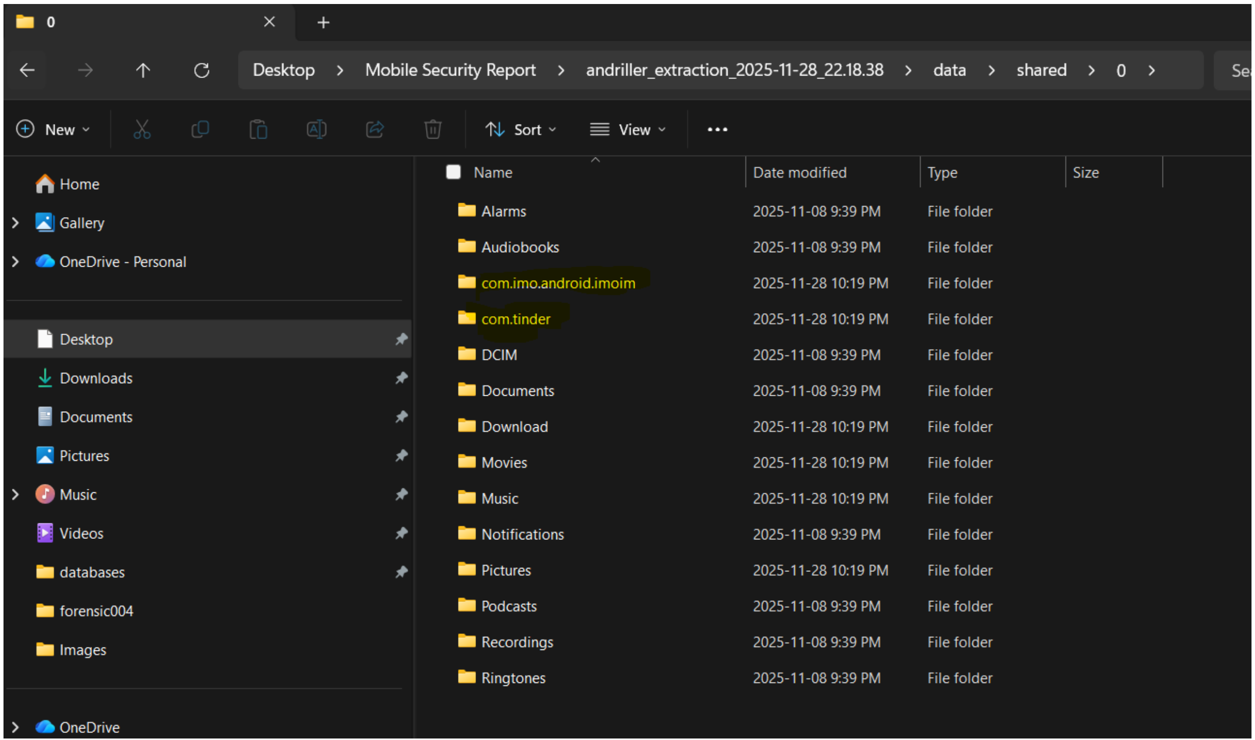


# This report was generated using Andriller CE # (This field is editable in Preferences)

[Andriller Report]

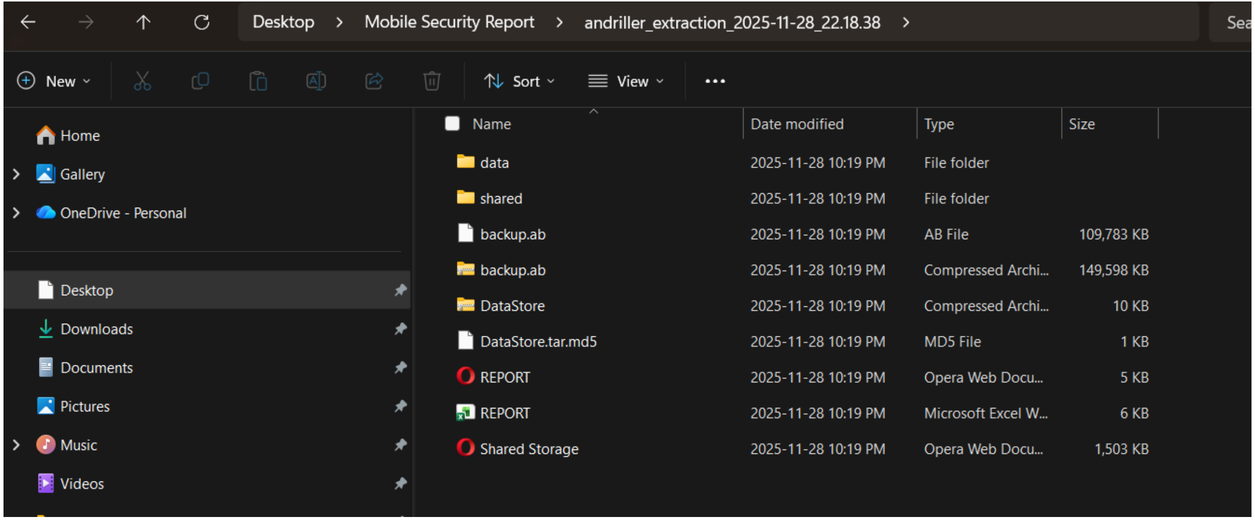
Type	Data
Serial	emulator-5554
Status	device
Permisson	root
Ro.Build.Version.Release	15
Ro.Build.Display.Id	sdk_gphone64_x86_64-userdebug 15 AE3A.240806.043 12960925 dev-keys
Wifi Mac	02:15:b2:00:00:00
Local_Time	2025-11-28 22:18:37 Pacific Standard Time
Device_Time	2025-11-28 22:18:37 PST
Accounts	• com.imo.android.imoim: imo • com.google: mobikeaecurity@gmail.com • com.reddit.account: Reddit for Android • com.reddit.account: Akunaatakasi
Application	Shared Storage (2572)

# andriller.com # (This field is editable in Preferences)



Desktop > Mobile Security Report > andriller_extraction_2025-11-28_22.18.38 > data > shared > 0 >

Name	Date modified	Type	Size
Alarms	2025-11-08 9:39 PM	File folder	
Audiobooks	2025-11-08 9:39 PM	File folder	
com.imo.android.imoim	2025-11-28 10:19 PM	File folder	
com.tinder	2025-11-28 10:19 PM	File folder	
DCIM	2025-11-08 9:39 PM	File folder	
Documents	2025-11-08 9:39 PM	File folder	
Download	2025-11-28 10:19 PM	File folder	
Movies	2025-11-28 10:19 PM	File folder	
Music	2025-11-28 10:19 PM	File folder	
Notifications	2025-11-08 9:39 PM	File folder	
Pictures	2025-11-28 10:19 PM	File folder	
Podcasts	2025-11-08 9:39 PM	File folder	
Recordings	2025-11-08 9:39 PM	File folder	
Ringtones	2025-11-08 9:39 PM	File folder	



Desktop > Mobile Security Report > andriller_extraction_2025-11-28_22.18.38 >

Name	Date modified	Type	Size
data	2025-11-28 10:19 PM	File folder	
shared	2025-11-28 10:19 PM	File folder	
backup.ab	2025-11-28 10:19 PM	AB File	109,783 KB
backup.ab	2025-11-28 10:19 PM	Compressed Archi...	149,598 KB
DataStore	2025-11-28 10:19 PM	Compressed Archi...	10 KB
DataStore.tar.md5	2025-11-28 10:19 PM	MD5 File	1 KB
REPORT	2025-11-28 10:19 PM	Opera Web Docu...	5 KB
REPORT	2025-11-28 10:19 PM	Microsoft Excel W...	6 KB
Shared Storage	2025-11-28 10:19 PM	Opera Web Docu...	1,503 KB

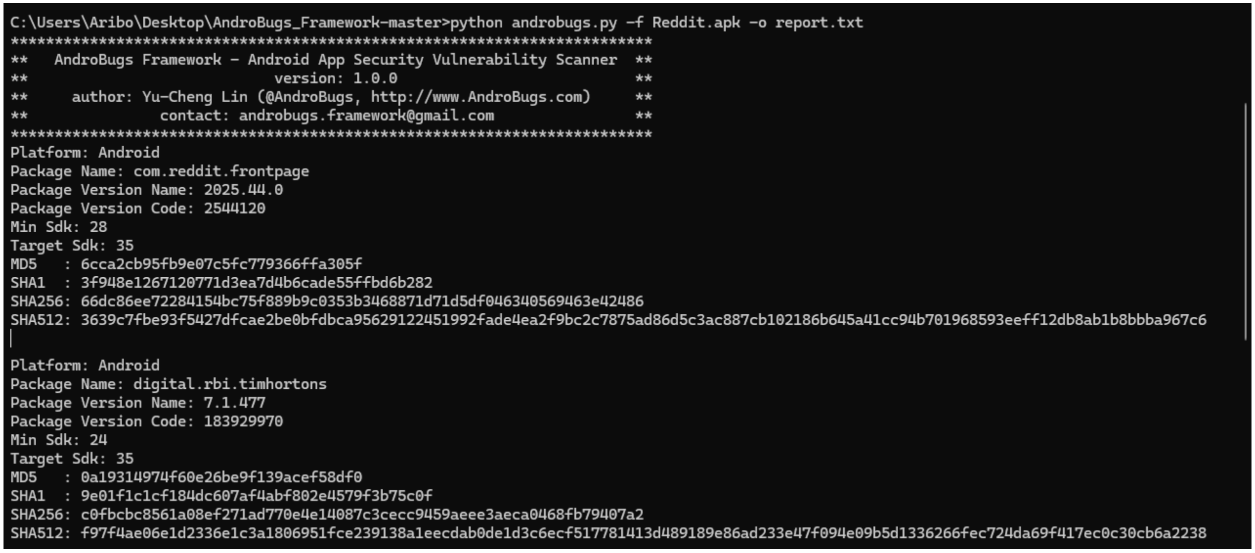
FOR TIMHORTONS AND REDDIT:

Phase 1: Static Analysis (Primary Analysis Phase)

Step 1: APK Acquisition and Preparation

- Download APK files directly from devices using ADB (Android Debug Bridge)
- Verify file integrity using MD5/SHA256 checksums

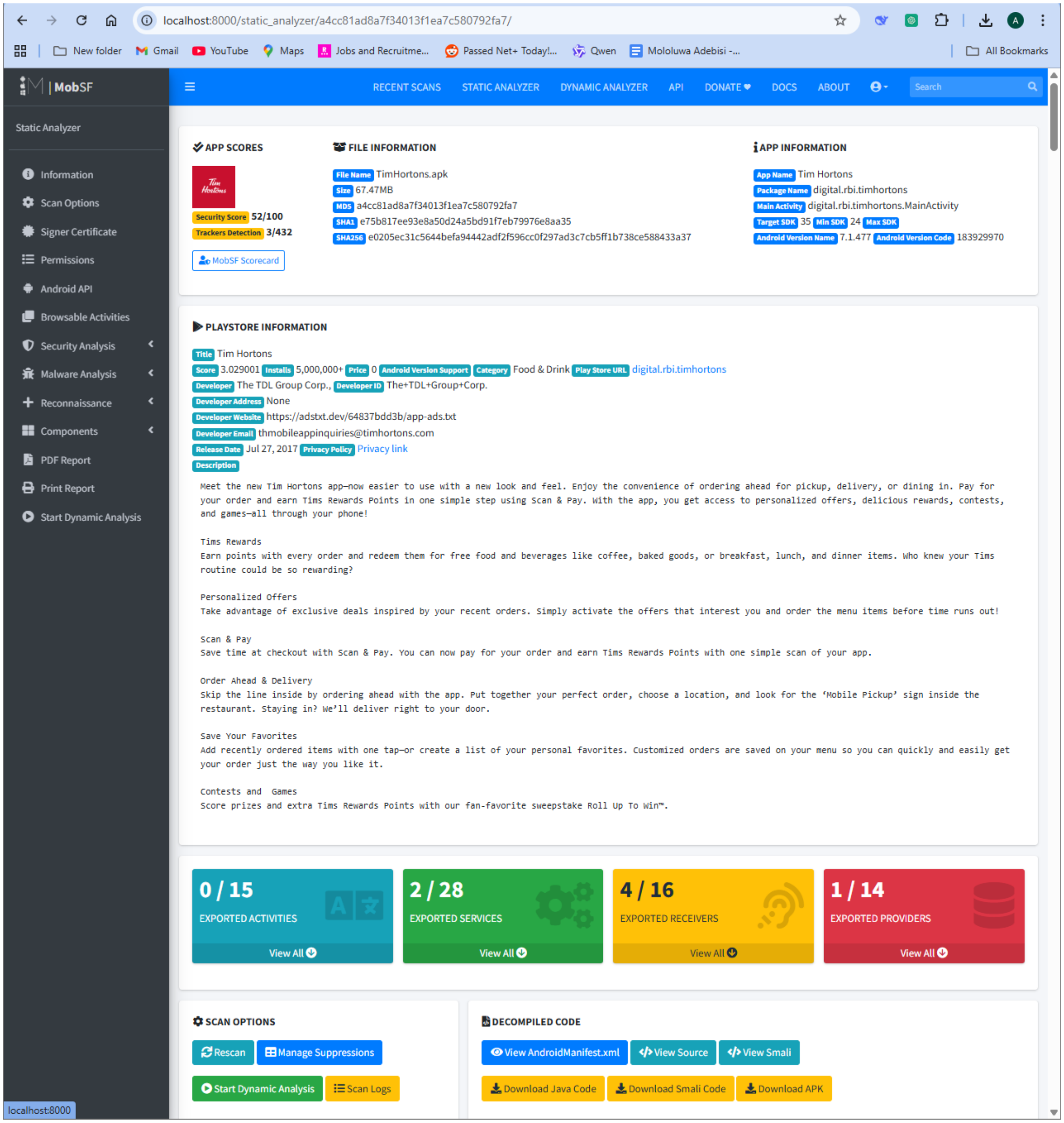
- Document application version, build number, and package name



```
C:\Users\Aribo\Desktop\AndroBugs_Framework-master>python androbugs.py -f Reddit.apk -o report.txt
*************************************************************************
**   AndroBugs Framework - Android App Security Vulnerability Scanner  **
**                            version: 1.0.0                           **
**     author: Yu-Cheng Lin (@AndroBugs, http://www.AndroBugs.com)     **
**               contact: androbugs.framework@gmail.com                **
*************************************************************************
Platform: Android
Package Name: com.reddit.frontpage
Package Version Name: 2025.44.0
Package Version Code: 2544120
Min Sdk: 28
Target Sdk: 35
MD5   : 6cca2cb95fb9e07c5fc779366ffa305f
SHA1  : 3f948e1267120771d3ea7d4b6cade55ffbd6b282
SHA256: 66dc86ee72284154bc75f889b9c0353b3468871d71d5df046340569463e42486
SHA512: 3639c7fbe93f5427dfcae2be0bfdbca95629122451992fade4ea2f9bc2c7875ad86d5c3ac887cb102186b645a41cc94b701968593eeff12db8ab1b8bbba967c6
|

Platform: Android
Package Name: digital.rbi.timhortons
Package Version Name: 7.1.477
Package Version Code: 183929970
Min Sdk: 24
Target Sdk: 35
MD5   : 0a19314974f60e26be9f139acef58df0
SHA1  : 9e01f1c1cf184dc607af4abf802e4579f3b75c0f
SHA256: c0fbcbc8561a08ef271ad770e4e14087c3cecc9459aeee3aeca0468fb79407a2
SHA512: f97f4ae06e1d2336e1c3a1806951fce239138a1eecdab0de1d3c6ecf517781413d489189e86ad233e47f094e09b5d1336266fec724da69f417ec0c30cb6a2238
```

Step 2: MobSF Static Analysis

- Upload APK to MobSF web interface
- Configure analysis parameters (enable all checks, deep scan mode)
- Execute comprehensive static scan
- Generate detailed security report
- Export findings in structured format (JSON, PDF)



APP SCORES

Security Score 52/100

Trackers Detection 3/432

FILE INFORMATION

File Name TimHortons.apk

Size 67.47MB

MD5 a4cc81ad8a7f34013f1ea7c580792fa7

SHA1 e75b817ee93e8a50d24a5bd91f7eb79976e8aa35

SHA256 e0205ec31c5644befa94442adf2f596cc0f297ad3c7cb5ff1b738ce588433a37

APP INFORMATION

App Name Tim Hortons

Package Name digital.rbi.timhortons

Main Activity digital.rbi.timhortons.MainActivity

Target SDK 35 Min SDK 24 Max SDK

Android Version Name 7.1.477 Android Version Code 183929970

PLAYSTORE INFORMATION

Title Tim Hortons

Score 3.029001 Installs 5,000,000+ Price 0 Android Version Support Category Food & Drink Play Store URL digital.rbi.timhortons

Developer The TDL Group Corp., Developer ID The+TDL+Group+Corp.

Developer Address None

Developer Website https://adstxt.dev/64837bdd3b/app-ads.txt

Developer Email thmobileappinquiries@timhortons.com

Release Date Jul 27, 2017 Privacy Policy Privacy link

Description

Meet the new Tim Hortons app—now easier to use with a new look and feel. Enjoy the convenience of ordering ahead for pickup, delivery, or dining in. Pay for your order and earn Tims Rewards Points in one simple step using Scan & Pay. With the app, you get access to personalized offers, delicious rewards, contests, and games—all through your phone!

Tims Rewards
Earn points with every order and redeem them for free food and beverages like coffee, baked goods, or breakfast, lunch, and dinner items. Who knew your Tims routine could be so rewarding?

Personalized Offers
Take advantage of exclusive deals inspired by your recent orders. Simply activate the offers that interest you and order the menu items before time runs out!

Scan & Pay
Save time at checkout with Scan & Pay. You can now pay for your order and earn Tims Rewards Points with one simple scan of your app.

Order Ahead & Delivery
Skip the line inside by ordering ahead with the app. Put together your perfect order, choose a location, and look for the 'Mobile Pickup' sign inside the restaurant. Staying in? We'll deliver right to your door.

Save Your Favorites
Add recently ordered items with one tap—or create a list of your personal favorites. Customized orders are saved on your menu so you can quickly and easily get your order just the way you like it.

Contests and Games
Score prizes and extra Tims Rewards Points with our fan-favorite sweepstake Roll Up To Win™.

0 / 15 EXPORTED ACTIVITIES

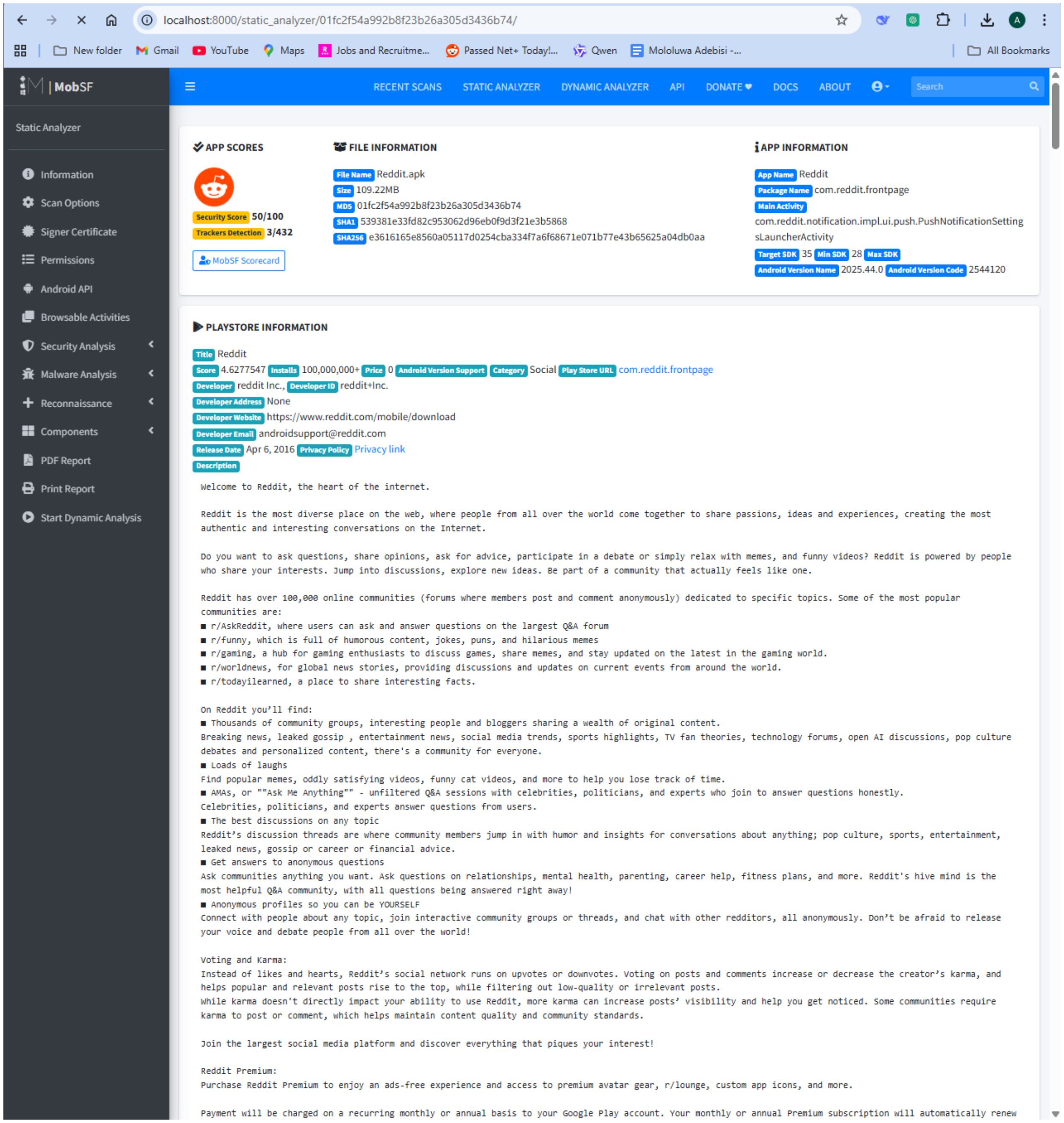
2 / 28 EXPORTED SERVICES

4 / 16 EXPORTED RECEIVERS

1 / 14 EXPORTED PROVIDERS

SCAN OPTIONS

DECOMPILED CODE



APP SCORES

Security Score 50/100
Trackers Detection 3/432

FILE INFORMATION

File Name Reddit.apk
Size 109.22MB
MD5 01fc2f54a992b8f23b26a305d3436b74
SHA1 539381e33fd82c953062d96eb0f9d3f21e3b5868
SHA256 e3616165e8560a05117d0254cba334f7a6f68671e071b77e43b65625a04db0aa

APP INFORMATION

App Name Reddit
Package Name com.reddit.frontpage
Main Activity
com.reddit.notification.impl.ui.push.PushNotificationSettingsLauncherActivity
Target SDK 35 Min SDK 28 Max SDK
Android Version Name 2025.44.0 Android Version Code 2544120

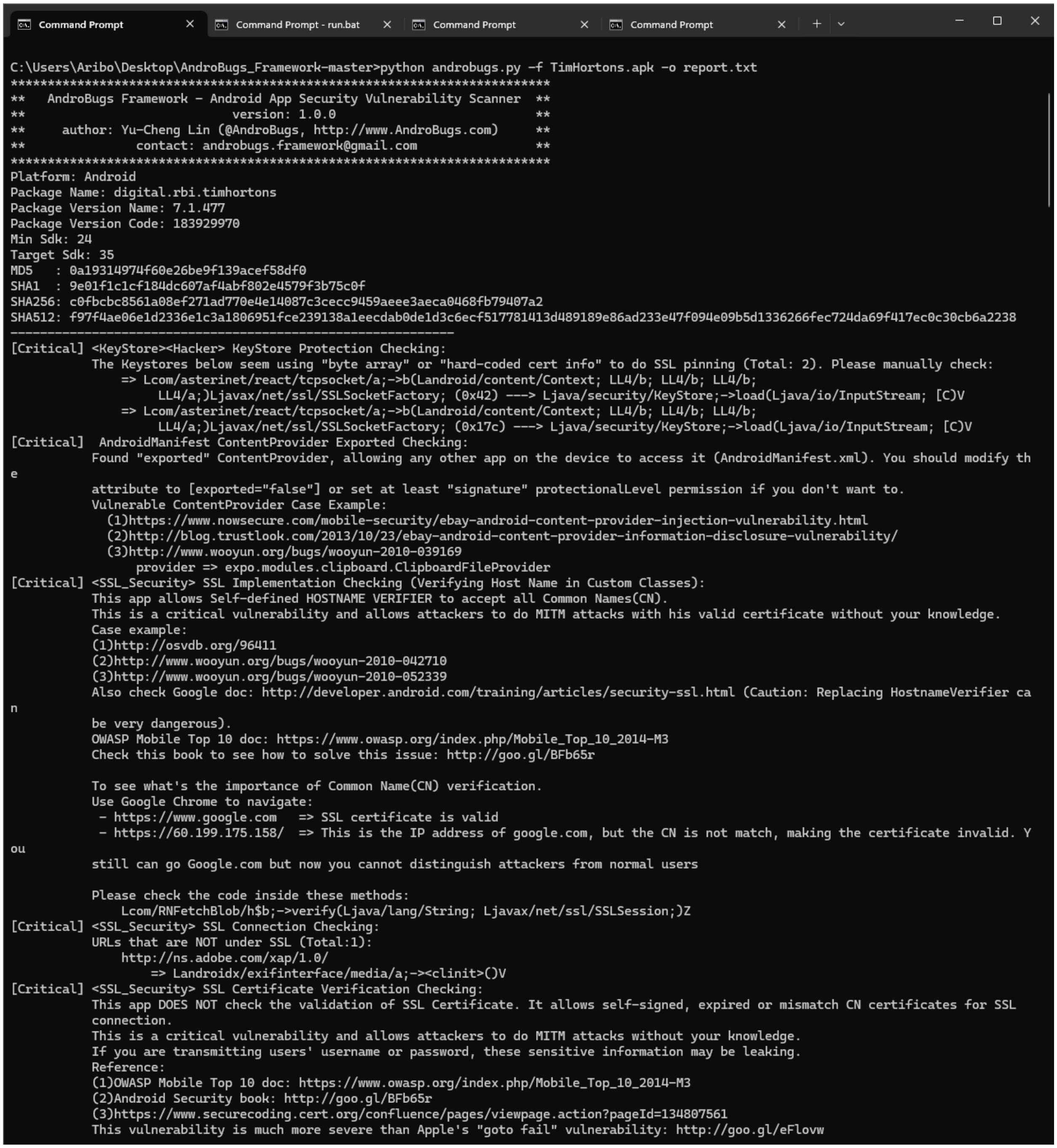
PLAYSTORE INFORMATION

Title Reddit
Score 4.6277547 Installs 100,000,000+ Price 0 Android Version Support Category Social Play Store URL com.reddit.frontpage
Developer reddit Inc., Developer ID reddit+Inc.
Developer Address None
Developer Website https://www.reddit.com/mobile/download
Developer Email androidsupport@reddit.com
Release Date Apr 6, 2016 Privacy Policy Privacy link
Description

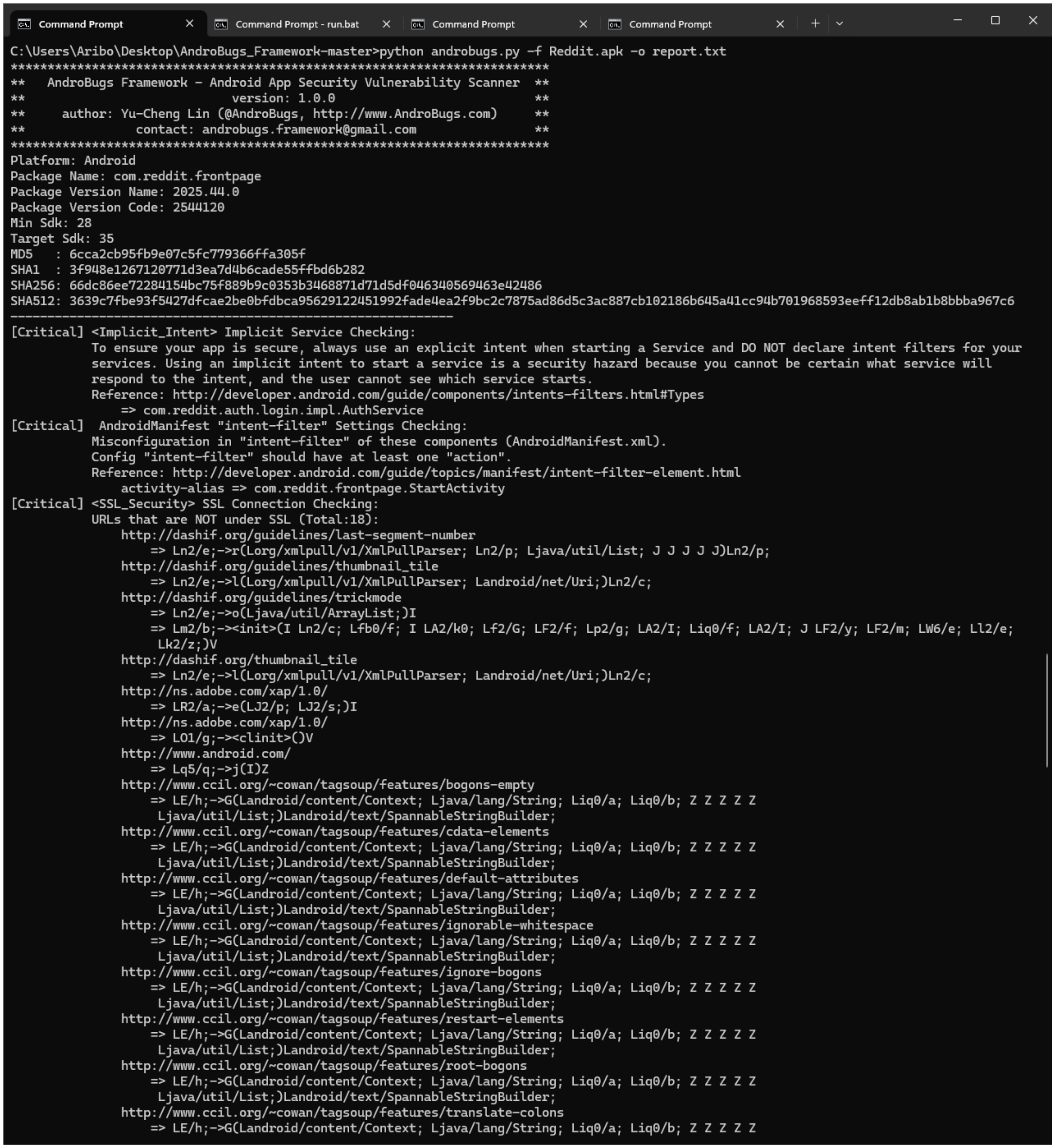
```
Welcome to Reddit, the heart of the internet.

Reddit is the most diverse place on the web, where people from all over the world come together to share passions, ideas and experiences, creating the most
authentic and interesting conversations on the Internet.

Do you want to ask questions, share opinions, ask for advice, participate in a debate or simply relax with memes, and funny videos? Reddit is powered by people
who share your interests. Jump into discussions, explore new ideas. Be part of a community that actually feels like one.

Reddit has over 100,000 online communities (forums where members post and comment anonymously) dedicated to specific topics. Some of the most popular
communities are:
■ r/AskReddit, where users can ask and answer questions on the largest Q&A forum
■ r/funny, which is full of humorous content, jokes, puns, and hilarious memes
■ r/gaming, a hub for gaming enthusiasts to discuss games, share memes, and stay updated on the latest in the gaming world.
■ r/worldnews, for global news stories, providing discussions and updates on current events from around the world.
■ r/todayilearned, a place to share interesting facts.

On Reddit you'll find:
■ Thousands of community groups, interesting people and bloggers sharing a wealth of original content.
Breaking news, leaked gossip , entertainment news, social media trends, sports highlights, TV fan theories, technology forums, open AI discussions, pop culture
debates and personalized content, there's a community for everyone.
■ Loads of laughs
Find popular memes, oddly satisfying videos, funny cat videos, and more to help you lose track of time.
■ AMAs, or ""Ask Me Anything"" - unfiltered Q&A sessions with celebrities, politicians, and experts who join to answer questions honestly.
Celebrities, politicians, and experts answer questions from users.
■ The best discussions on any topic
Reddit's discussion threads are where community members jump in with humor and insights for conversations about anything; pop culture, sports, entertainment,
leaked news, gossip or career or financial advice.
■ Get answers to anonymous questions
Ask communities anything you want. Ask questions on relationships, mental health, parenting, career help, fitness plans, and more. Reddit's hive mind is the
most helpful Q&A community, with all questions being answered right away!
■ Anonymous profiles so you can be YOURSELF
Connect with people about any topic, join interactive community groups or threads, and chat with other redditors, all anonymously. Don't be afraid to release
your voice and debate people from all over the world!

Voting and Karma:
Instead of likes and hearts, Reddit's social network runs on upvotes or downvotes. Voting on posts and comments increase or decrease the creator's karma, and
helps popular and relevant posts rise to the top, while filtering out low-quality or irrelevant posts.
While karma doesn't directly impact your ability to use Reddit, more karma can increase posts' visibility and help you get noticed. Some communities require
karma to post or comment, which helps maintain content quality and community standards.

Join the largest social media platform and discover everything that piques your interest!

Reddit Premium:
Purchase Reddit Premium to enjoy an ads-free experience and access to premium avatar gear, r/lounge, custom app icons, and more.

Payment will be charged on a recurring monthly or annual basis to your Google Play account. Your monthly or annual Premium subscription will automatically renew
```

Step 3: AndroBugs Framework Analysis

- Execute AndroBugs command-line analysis: python androbugs.py -f [APK_FILE]
- Configure analysis depth and module selection
- Process output and generate vulnerability report
- Cross-reference findings with MobSF results



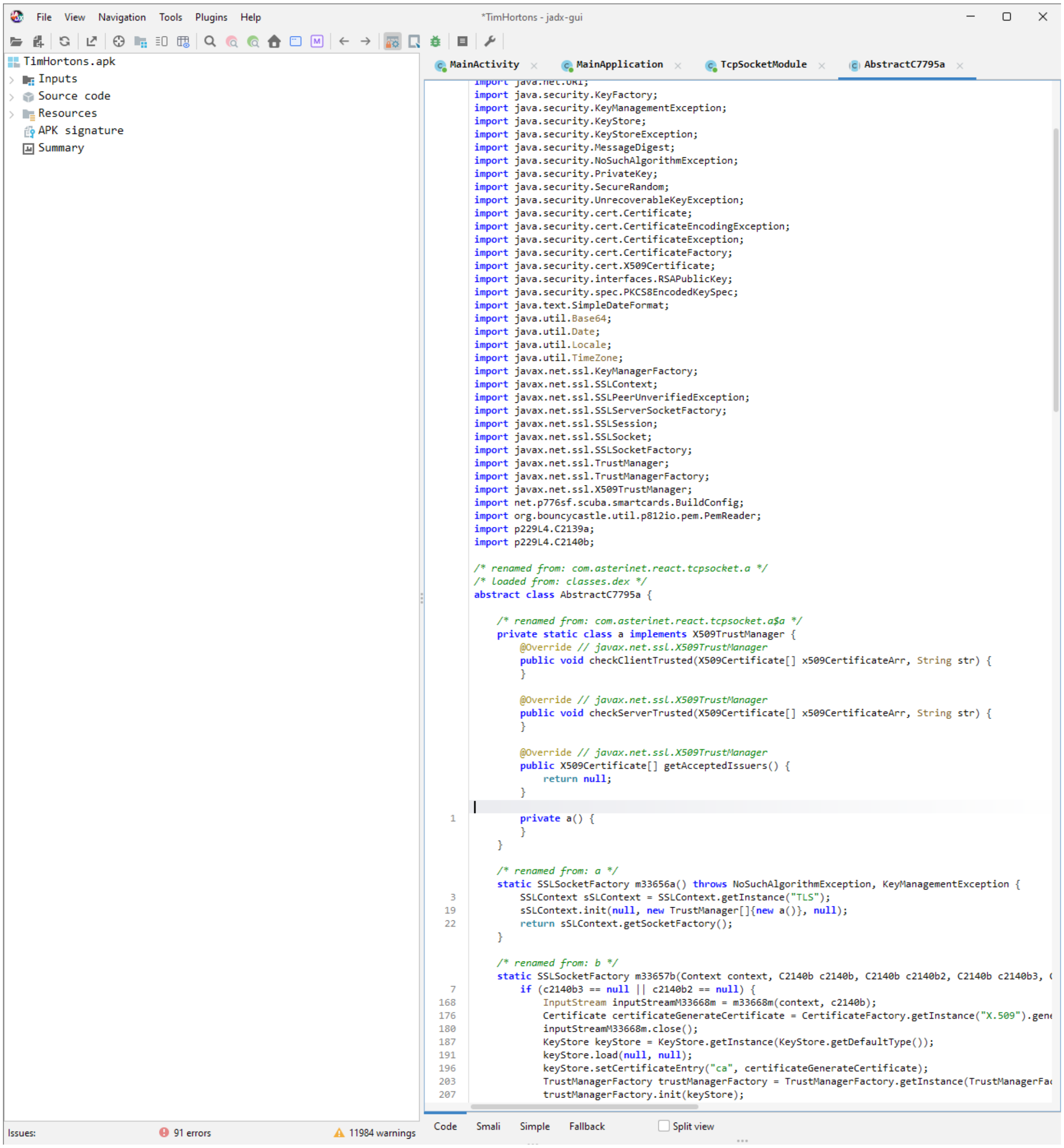
```
C:\Users\Aribo\Desktop\AndroBugs_Framework-master>python androbugs.py -f TimHortons.apk -o report.txt
*************************************************************************
**   AndroBugs Framework - Android App Security Vulnerability Scanner  **
**                            version: 1.0.0                           **
**     author: Yu-Cheng Lin (@AndroBugs, http://www.AndroBugs.com)     **
**               contact: androbugs.framework@gmail.com                **
*************************************************************************
Platform: Android
Package Name: digital.rbi.timhortons
Package Version Name: 7.1.477
Package Version Code: 183929970
Min Sdk: 24
Target Sdk: 35
MD5   : 0a19314974f60e26be9f139acef58df0
SHA1  : 9e01f1c1cf184dc607af4abf802e4579f3b75c0f
SHA256: c0fbcbc8561a08ef271ad770e4e14087c3cecc9459aeee3aeca0468fb79407a2
SHA512: f97f4ae06e1d2336e1c3a1806951fce239138a1eecdab0de1d3c6ecf517781413d489189e86ad233e47f094e09b5d1336266fec724da69f417ec0c30cb6a2238
------------------------------------------------------------
[Critical] <KeyStore><Hacker> KeyStore Protection Checking:
           The Keystores below seem using "byte array" or "hard-coded cert info" to do SSL pinning (Total: 2). Please manually check:
               => Lcom/asterinet/react/tcpsocket/a;->b(Landroid/content/Context; LL4/b; LL4/b; LL4/b;
                    LL4/a;)Ljavax/net/ssl/SSLSocketFactory; (0x42) ---> Ljava/security/KeyStore;->load(Ljava/io/InputStream; [C)V
               => Lcom/asterinet/react/tcpsocket/a;->b(Landroid/content/Context; LL4/b; LL4/b; LL4/b;
                    LL4/a;)Ljavax/net/ssl/SSLSocketFactory; (0x17c) ---> Ljava/security/KeyStore;->load(Ljava/io/InputStream; [C)V
[Critical]  AndroidManifest ContentProvider Exported Checking:
           Found "exported" ContentProvider, allowing any other app on the device to access it (AndroidManifest.xml). You should modify th
e
           attribute to [exported="false"] or set at least "signature" protectionalLevel permission if you don't want to.
           Vulnerable ContentProvider Case Example:
             (1)https://www.nowsecure.com/mobile-security/ebay-android-content-provider-injection-vulnerability.html
             (2)http://blog.trustlook.com/2013/10/23/ebay-android-content-provider-information-disclosure-vulnerability/
             (3)http://www.wooyun.org/bugs/wooyun-2010-039169
                 provider => expo.modules.clipboard.ClipboardFileProvider
[Critical] <SSL_Security> SSL Implementation Checking (Verifying Host Name in Custom Classes):
           This app allows Self-defined HOSTNAME VERIFIER to accept all Common Names(CN).
           This is a critical vulnerability and allows attackers to do MITM attacks with his valid certificate without your knowledge.
           Case example:
           (1)http://osvdb.org/96411
           (2)http://www.wooyun.org/bugs/wooyun-2010-042710
           (3)http://www.wooyun.org/bugs/wooyun-2010-052339
           Also check Google doc: http://developer.android.com/training/articles/security-ssl.html (Caution: Replacing HostnameVerifier ca
n
           be very dangerous).
           OWASP Mobile Top 10 doc: https://www.owasp.org/index.php/Mobile_Top_10_2014-M3
           Check this book to see how to solve this issue: http://goo.gl/BFb65r

           To see what's the importance of Common Name(CN) verification.
           Use Google Chrome to navigate:
            - https://www.google.com   => SSL certificate is valid
            - https://60.199.175.158/  => This is the IP address of google.com, but the CN is not match, making the certificate invalid. Y
ou
           still can go Google.com but now you cannot distinguish attackers from normal users

           Please check the code inside these methods:
               Lcom/RNFetchBlob/h$b;->verify(Ljava/lang/String; Ljavax/net/ssl/SSLSession;)Z
[Critical] <SSL_Security> SSL Connection Checking:
           URLs that are NOT under SSL (Total:1):
               http://ns.adobe.com/xap/1.0/
                   => Landroidx/exifinterface/media/a;-><clinit>()V
[Critical] <SSL_Security> SSL Certificate Verification Checking:
           This app DOES NOT check the validation of SSL Certificate. It allows self-signed, expired or mismatch CN certificates for SSL
           connection.
           This is a critical vulnerability and allows attackers to do MITM attacks without your knowledge.
           If you are transmitting users' username or password, these sensitive information may be leaking.
           Reference:
           (1)OWASP Mobile Top 10 doc: https://www.owasp.org/index.php/Mobile_Top_10_2014-M3
           (2)Android Security book: http://goo.gl/BFb65r
           (3)https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=134807561
           This vulnerability is much more severe than Apple's "goto fail" vulnerability: http://goo.gl/eFlovw
```



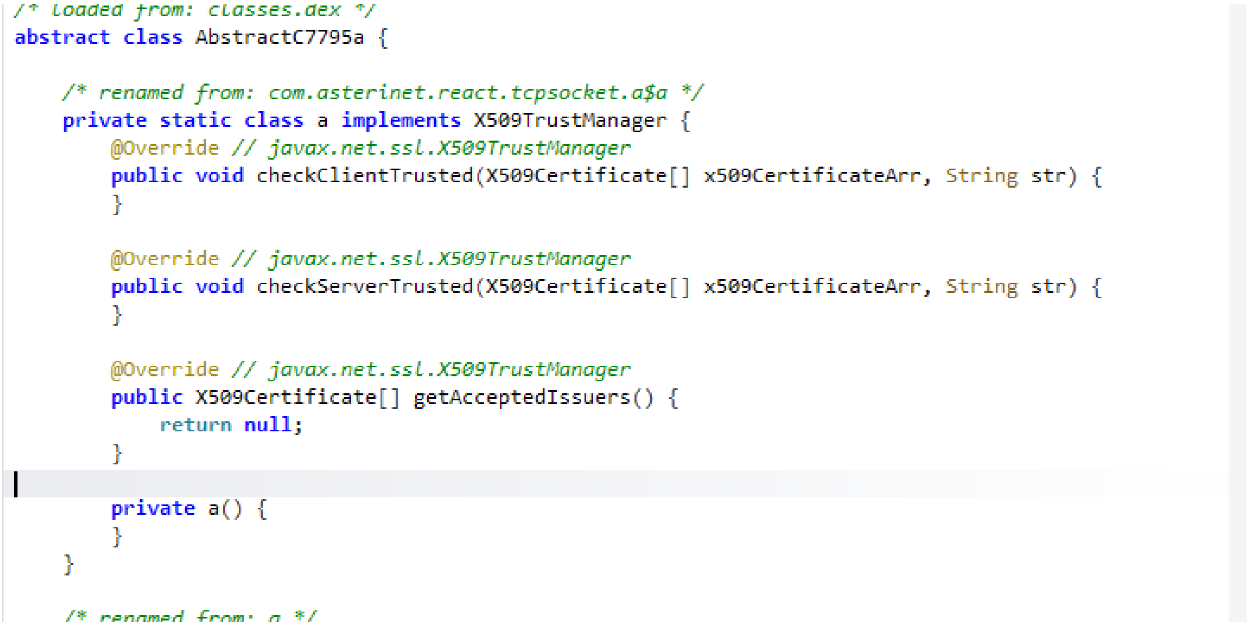
```
C:\Users\Aribo\Desktop\AndroBugs_Framework-master>python androbugs.py -f Reddit.apk -o report.txt
*************************************************************************
**   AndroBugs Framework - Android App Security Vulnerability Scanner  **
**                            version: 1.0.0                           **
**     author: Yu-Cheng Lin (@AndroBugs, http://www.AndroBugs.com)     **
**               contact: androbugs.framework@gmail.com                **
*************************************************************************
Platform: Android
Package Name: com.reddit.frontpage
Package Version Name: 2025.44.0
Package Version Code: 2544120
Min Sdk: 28
Target Sdk: 35
MD5   : 6cca2cb95fb9e07c5fc779366ffa305f
SHA1  : 3f948e1267120771d3ea7d4b6cade55ffbd6b282
SHA256: 66dc86ee72284154bc75f889b9c0353b3468871d71d5df046340569463e42486
SHA512: 3639c7fbe93f5427dfcae2be0bfdbca95629122451992fade4ea2f9bc2c7875ad86d5c3ac887cb102186b645a41cc94b701968593eeff12db8ab1b8bbba967c6
------------------------------------------------------------
[Critical] <Implicit_Intent> Implicit Service Checking:
           To ensure your app is secure, always use an explicit intent when starting a Service and DO NOT declare intent filters for your
           services. Using an implicit intent to start a service is a security hazard because you cannot be certain what service will
           respond to the intent, and the user cannot see which service starts.
           Reference: http://developer.android.com/guide/components/intents-filters.html#Types
               => com.reddit.auth.login.impl.AuthService
[Critical]  AndroidManifest "intent-filter" Settings Checking:
           Misconfiguration in "intent-filter" of these components (AndroidManifest.xml).
           Config "intent-filter" should have at least one "action".
           Reference: http://developer.android.com/guide/topics/manifest/intent-filter-element.html
               activity-alias => com.reddit.frontpage.StartActivity
[Critical] <SSL_Security> SSL Connection Checking:
           URLs that are NOT under SSL (Total:18):
               http://dashif.org/guidelines/last-segment-number
                   => Ln2/e;->r(Lorg/xmlpull/v1/XmlPullParser; Ln2/p; Ljava/util/List; J J J J J)Ln2/p;
               http://dashif.org/guidelines/thumbnail_tile
                   => Ln2/e;->l(Lorg/xmlpull/v1/XmlPullParser; Landroid/net/Uri;)Ln2/c;
               http://dashif.org/guidelines/trickmode
                   => Ln2/e;->o(Ljava/util/ArrayList;)I
                   => Lm2/b;-><init>(I Ln2/c; Lfb0/f; I LA2/k0; Lf2/G; LF2/f; Lp2/g; LA2/I; Liq0/f; LA2/I; J LF2/y; LF2/m; LW6/e; Ll2/e;
                    Lk2/z;)V
               http://dashif.org/thumbnail_tile
                   => Ln2/e;->l(Lorg/xmlpull/v1/XmlPullParser; Landroid/net/Uri;)Ln2/c;
               http://ns.adobe.com/xap/1.0/
                   => LR2/a;->e(LJ2/p; LJ2/s;)I
               http://ns.adobe.com/xap/1.0/
                   => LO1/g;-><clinit>()V
               http://www.android.com/
                   => Lq5/q;->j(I)Z
               http://www.ccil.org/~cowan/tagsoup/features/bogons-empty
                   => LE/h;->G(Landroid/content/Context; Ljava/lang/String; Liq0/a; Liq0/b; Z Z Z Z Z
                    Ljava/util/List;)Landroid/text/SpannableStringBuilder;
               http://www.ccil.org/~cowan/tagsoup/features/cdata-elements
                   => LE/h;->G(Landroid/content/Context; Ljava/lang/String; Liq0/a; Liq0/b; Z Z Z Z Z
                    Ljava/util/List;)Landroid/text/SpannableStringBuilder;
               http://www.ccil.org/~cowan/tagsoup/features/default-attributes
                   => LE/h;->G(Landroid/content/Context; Ljava/lang/String; Liq0/a; Liq0/b; Z Z Z Z Z
                    Ljava/util/List;)Landroid/text/SpannableStringBuilder;
               http://www.ccil.org/~cowan/tagsoup/features/ignorable-whitespace
                   => LE/h;->G(Landroid/content/Context; Ljava/lang/String; Liq0/a; Liq0/b; Z Z Z Z Z
                    Ljava/util/List;)Landroid/text/SpannableStringBuilder;
               http://www.ccil.org/~cowan/tagsoup/features/ignore-bogons
                   => LE/h;->G(Landroid/content/Context; Ljava/lang/String; Liq0/a; Liq0/b; Z Z Z Z Z
                    Ljava/util/List;)Landroid/text/SpannableStringBuilder;
               http://www.ccil.org/~cowan/tagsoup/features/restart-elements
                   => LE/h;->G(Landroid/content/Context; Ljava/lang/String; Liq0/a; Liq0/b; Z Z Z Z Z
                    Ljava/util/List;)Landroid/text/SpannableStringBuilder;
               http://www.ccil.org/~cowan/tagsoup/features/root-bogons
                   => LE/h;->G(Landroid/content/Context; Ljava/lang/String; Liq0/a; Liq0/b; Z Z Z Z Z
                    Ljava/util/List;)Landroid/text/SpannableStringBuilder;
               http://www.ccil.org/~cowan/tagsoup/features/translate-colons
                   => LE/h;->G(Landroid/content/Context; Ljava/lang/String; Liq0/a; Liq0/b; Z Z Z Z Z
```

Step 4: Manual Code Review

- Decompile APK using jadx-gui for visual code inspection
- Focus on areas flagged by automated tools
- Examine critical security functions (authentication, encryption, network communication)
- Document code-level vulnerabilities with specific file and line references



```java
import java.net.URI;
import java.security.KeyFactory;
import java.security.KeyManagementException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.SecureRandom;
import java.security.UnrecoverableKeyException;
import java.security.cert.Certificate;
import java.security.cert.CertificateEncodingException;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.security.interfaces.RSAPublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.text.SimpleDateFormat;
import java.util.Base64;
import java.util.Date;
import java.util.Locale;
import java.util.TimeZone;
import javax.net.ssl.KeyManagerFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLPeerUnverifiedException;
import javax.net.ssl.SSLServerSocketFactory;
import javax.net.ssl.SSLSession;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.net.ssl.X509TrustManager;
import net.p776sf.scuba.smartcards.BuildConfig;
import org.bouncycastle.util.p812io.pem.PemReader;
import p229L4.C2139a;
import p229L4.C2140b;

/* renamed from: com.asterinet.react.tcpsocket.a */
/* loaded from: classes.dex */
abstract class AbstractC7795a {

    /* renamed from: com.asterinet.react.tcpsocket.a$a */
    private static class a implements X509TrustManager {
        @Override // javax.net.ssl.X509TrustManager
        public void checkClientTrusted(X509Certificate[] x509CertificateArr, String str) {
        }

        @Override // javax.net.ssl.X509TrustManager
        public void checkServerTrusted(X509Certificate[] x509CertificateArr, String str) {
        }

        @Override // javax.net.ssl.X509TrustManager
        public X509Certificate[] getAcceptedIssuers() {
            return null;
        }

        private a() {
        }
    }

    /* renamed from: a */
    static SSLSocketFactory m33656a() throws NoSuchAlgorithmException, KeyManagementException {
        SSLContext sSLContext = SSLContext.getInstance("TLS");
        sSLContext.init(null, new TrustManager[]{new a()}, null);
        return sSLContext.getSocketFactory();
    }

    /* renamed from: b */
    static SSLSocketFactory m33657b(Context context, C2140b c2140b, C2140b c2140b2, C2140b c2140b3, (
        if (c2140b3 == null || c2140b2 == null) {
            InputStream inputStreamM33668m = m33668m(context, c2140b);
            Certificate certificateGenerateCertificate = CertificateFactory.getInstance("X.509").gene
            inputStreamM33668m.close();
            KeyStore keyStore = KeyStore.getInstance(KeyStore.getDefaultType());
            keyStore.load(null, null);
            keyStore.setCertificateEntry("ca", certificateGenerateCertificate);
            TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance(TrustManagerFa
            trustManagerFactory.init(keyStore);
```



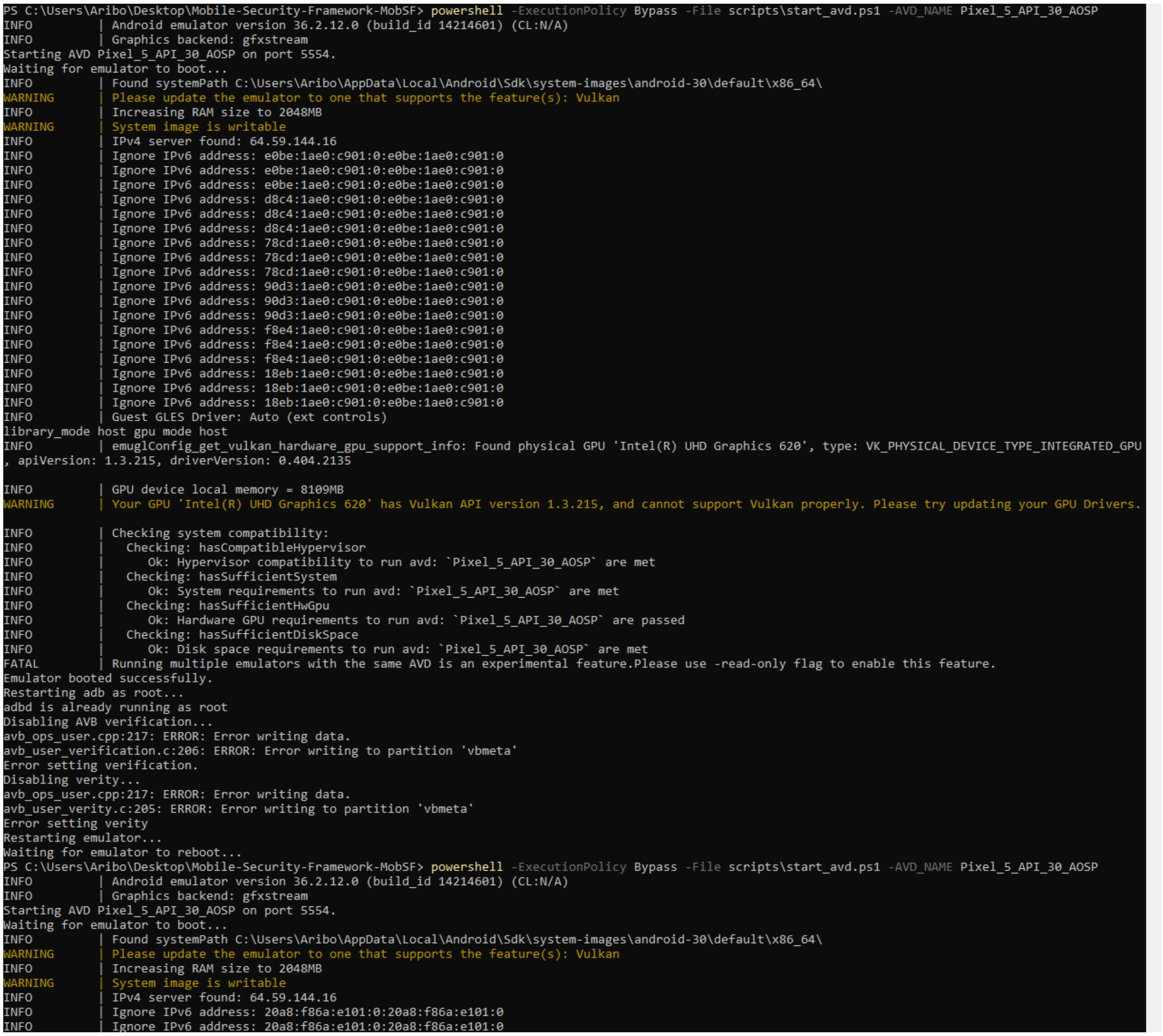
```
/* loaded from: classes.dex */
abstract class AbstractC7795a {

    /* renamed from: com.asterinet.react.tcpsocket.a$a */
    private static class a implements X509TrustManager {
        @Override // javax.net.ssl.X509TrustManager
        public void checkClientTrusted(X509Certificate[] x509CertificateArr, String str) {
        }

        @Override // javax.net.ssl.X509TrustManager
        public void checkServerTrusted(X509Certificate[] x509CertificateArr, String str) {
        }

        @Override // javax.net.ssl.X509TrustManager
        public X509Certificate[] getAcceptedIssuers() {
            return null;
        }

        private a() {
        }
    }

    /* renamed from: a */
```

Phase 2: Dynamic Analysis

Step 1: Environment Configuration

- Configure Android emulator (Android API 30 due to compatibility issues)

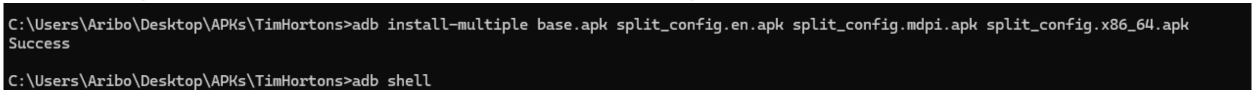
- Launch the emulator with:



```
PS C:\Users\Aribo\Desktop\Mobile-Security-Framework-MobSF> powershell -ExecutionPolicy Bypass -File scripts\start_avd.ps1 -AVD_NAME Pixel_5_API_30_AOSP
INFO         | Android emulator version 36.2.12.0 (build_id 14214601) (CL:N/A)
INFO         | Graphics backend: gfxstream
Starting AVD Pixel_5_API_30_AOSP on port 5554.
Waiting for emulator to boot...
INFO         | Found systemPath C:\Users\Aribo\AppData\Local\Android\Sdk\system-images\android-30\default\x86_64\
WARNING      | Please update the emulator to one that supports the feature(s): Vulkan
INFO         | Increasing RAM size to 2048MB
WARNING      | System image is writable
INFO         | IPv4 server found: 64.59.144.16
INFO         | Ignore IPv6 address: e0be:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: e0be:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: e0be:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: d8c4:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: d8c4:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: d8c4:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: 78cd:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: 78cd:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: 78cd:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: 90d3:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: 90d3:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: 90d3:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: f8e4:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: f8e4:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: f8e4:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: 18eb:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: 18eb:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Ignore IPv6 address: 18eb:1ae0:c901:0:e0be:1ae0:c901:0
INFO         | Guest GLES Driver: Auto (ext controls)
library_mode host gpu mode host
INFO         | emuglConfig_get_vulkan_hardware_gpu_support_info: Found physical GPU 'Intel(R) UHD Graphics 620', type: VK_PHYSICAL_DEVICE_TYPE_INTEGRATED_GPU
, apiVersion: 1.3.215, driverVersion: 0.404.2135

INFO         | GPU device local memory = 8109MB
WARNING      | Your GPU 'Intel(R) UHD Graphics 620' has Vulkan API version 1.3.215, and cannot support Vulkan properly. Please try updating your GPU Drivers.

INFO         | Checking system compatibility:
INFO         |   Checking: hasCompatibleHypervisor
INFO         |      Ok: Hypervisor compatibility to run avd: `Pixel_5_API_30_AOSP` are met
INFO         |   Checking: hasSufficientSystem
INFO         |      Ok: System requirements to run avd: `Pixel_5_API_30_AOSP` are met
INFO         |   Checking: hasSufficientHwGpu
INFO         |      Ok: Hardware GPU requirements to run avd: `Pixel_5_API_30_AOSP` are passed
INFO         |   Checking: hasSufficientDiskSpace
INFO         |      Ok: Disk space requirements to run avd: `Pixel_5_API_30_AOSP` are met
FATAL        | Running multiple emulators with the same AVD is an experimental feature.Please use -read-only flag to enable this feature.
Emulator booted successfully.
Restarting adb as root...
adbd is already running as root
Disabling AVB verification...
avb_ops_user.cpp:217: ERROR: Error writing data.
avb_user_verification.c:206: ERROR: Error writing to partition 'vbmeta'
Error setting verification.
Disabling verity...
avb_ops_user.cpp:217: ERROR: Error writing data.
avb_user_verity.c:205: ERROR: Error writing to partition 'vbmeta'
Error setting verity
Restarting emulator...
Waiting for emulator to reboot...
PS C:\Users\Aribo\Desktop\Mobile-Security-Framework-MobSF> powershell -ExecutionPolicy Bypass -File scripts\start_avd.ps1 -AVD_NAME Pixel_5_API_30_AOSP
INFO         | Android emulator version 36.2.12.0 (build_id 14214601) (CL:N/A)
INFO         | Graphics backend: gfxstream
Starting AVD Pixel_5_API_30_AOSP on port 5554.
Waiting for emulator to boot...
INFO         | Found systemPath C:\Users\Aribo\AppData\Local\Android\Sdk\system-images\android-30\default\x86_64\
WARNING      | Please update the emulator to one that supports the feature(s): Vulkan
INFO         | Increasing RAM size to 2048MB
WARNING      | System image is writable
INFO         | IPv4 server found: 64.59.144.16
INFO         | Ignore IPv6 address: 20a8:f86a:e101:0:20a8:f86a:e101:0
INFO         | Ignore IPv6 address: 20a8:f86a:e101:0:20a8:f86a:e101:0
```

Step 2: Application Installation and Initial Execution

- Install APK via ADB: adb install Reddit.apk/TimHortons.apk
- Launch application and complete initial setup
- Create test accounts with controlled credentials
- Begin traffic capture and system monitoring



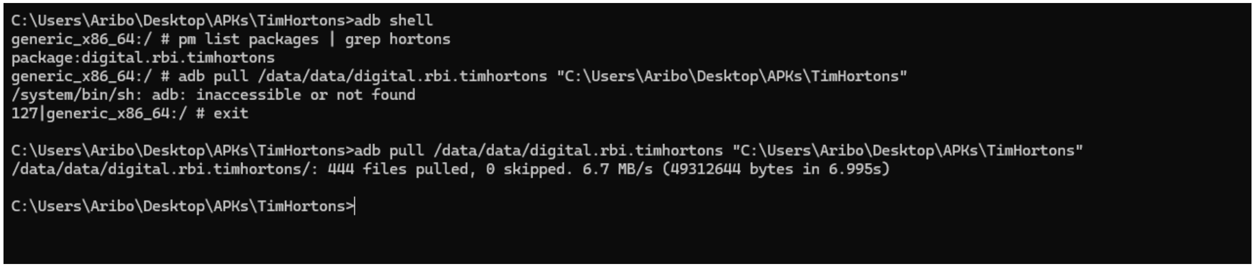
```
C:\Users\Aribo\Desktop\APKs\TimHortons>adb install-multiple base.apk split_config.en.apk split_config.mdpi.apk split_config.x86_64.apk
Success

C:\Users\Aribo\Desktop\APKs\TimHortons>adb shell
```

Step 3: Behavioral Testing

- Systematic feature exploration (registration, login, content browsing, transactions)
- Exercise all major application workflows
- Trigger permission requests and observe behavior
- Interact with exported components using intent fuzzing
- Monitor network traffic patterns and API communications

Step 4: Data Storage Analysis

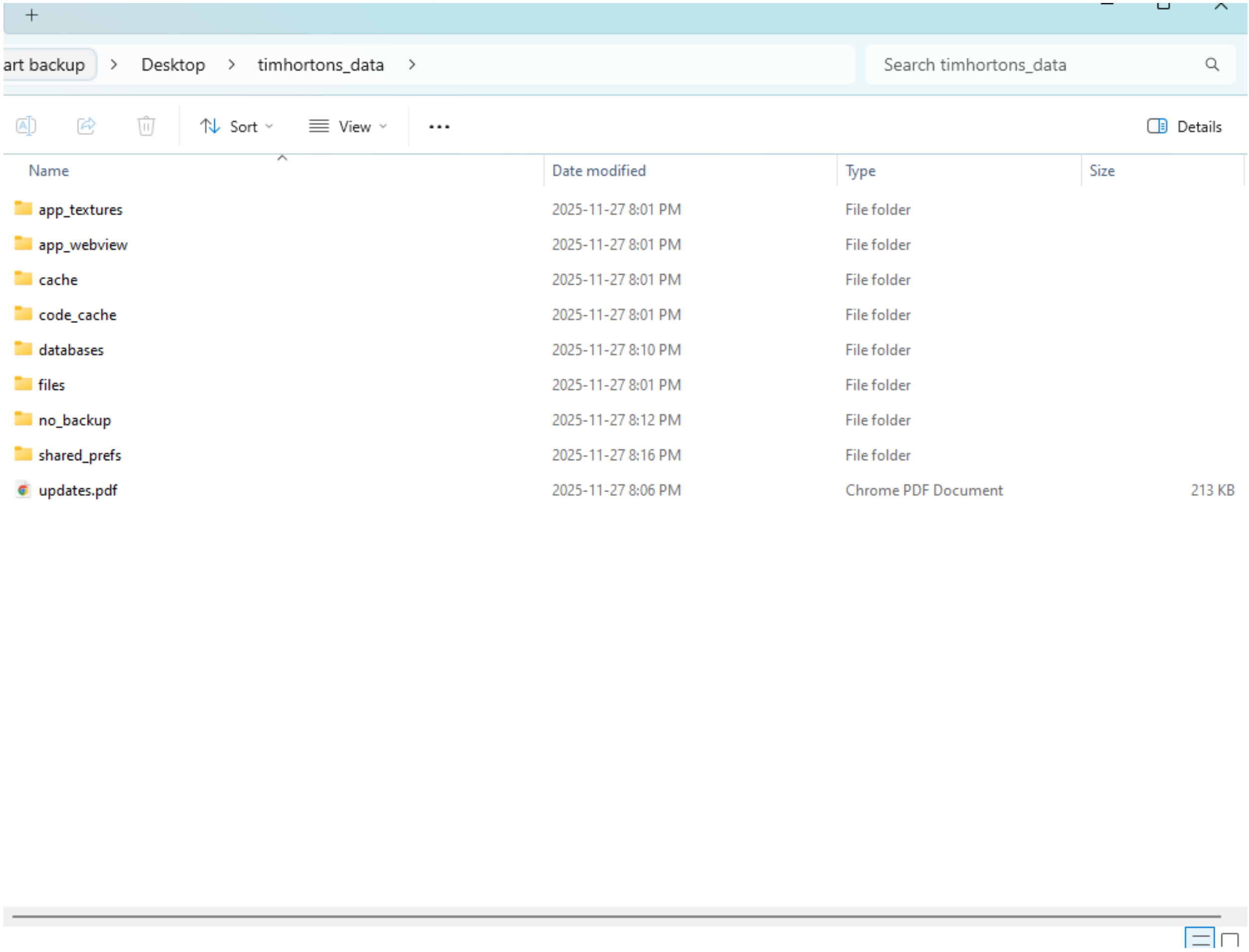
- Extract application data directory: adb pull /data/data/



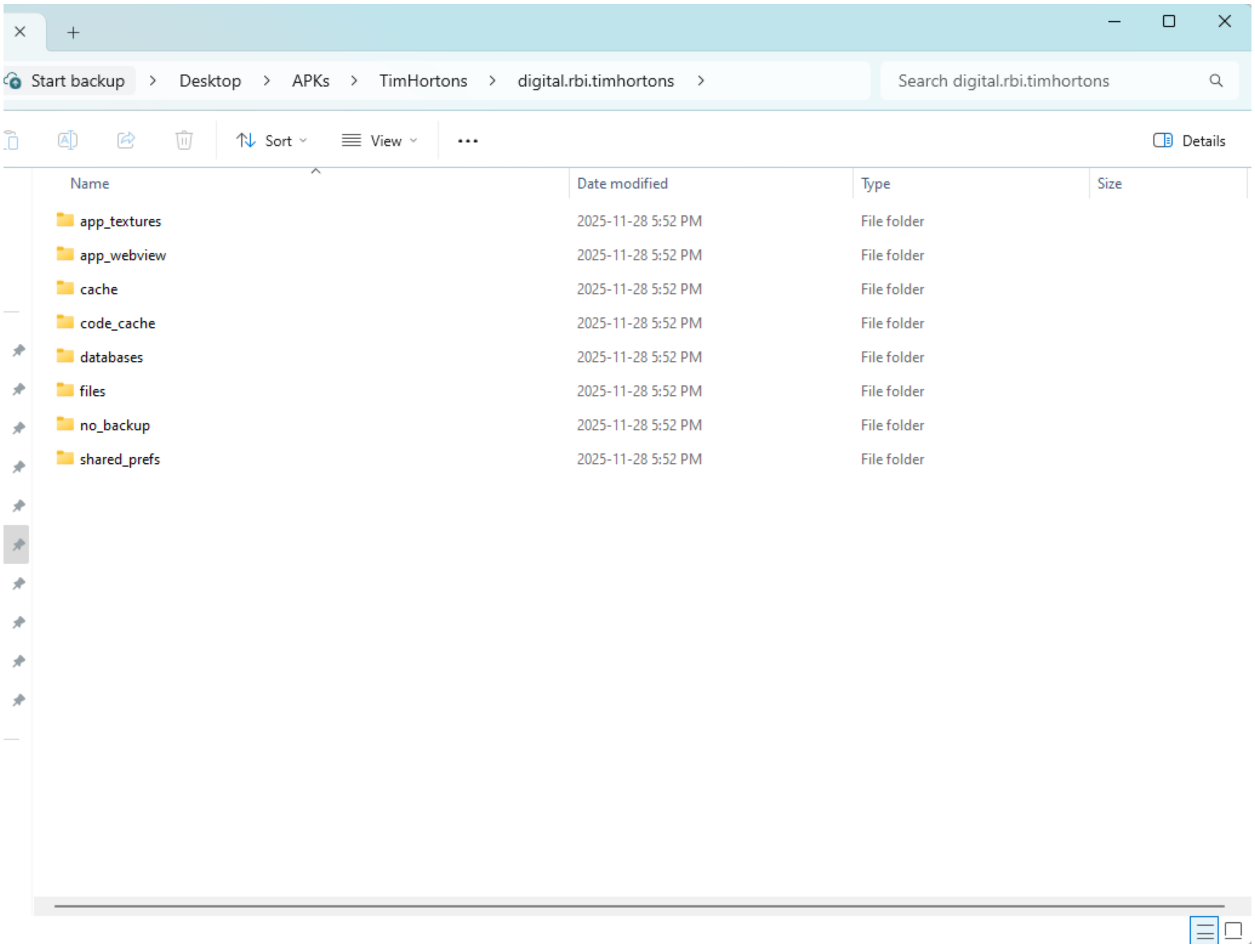
```
C:\Users\Aribo\Desktop\APKs\TimHortons>adb shell
generic_x86_64:/ # pm list packages | grep hortons
package:digital.rbi.timhortons
generic_x86_64:/ # adb pull /data/data/digital.rbi.timhortons "C:\Users\Aribo\Desktop\APKs\TimHortons"
/system/bin/sh: adb: inaccessible or not found
127|generic_x86_64:/ # exit

C:\Users\Aribo\Desktop\APKs\TimHortons>adb pull /data/data/digital.rbi.timhortons "C:\Users\Aribo\Desktop\APKs\TimHortons"
/data/data/digital.rbi.timhortons/: 444 files pulled, 0 skipped. 6.7 MB/s (49312644 bytes in 6.995s)

C:\Users\Aribo\Desktop\APKs\TimHortons>
```

- Analyzez SQLite databases using DB Browser for SQLite
- Examined SharedPreferences XML files
- Inspected cache directories and temporary files

- Document sensitive data storage practices



Name	Date modified	Type	Size
app_textures	2025-11-27 8:01 PM	File folder	
app_webview	2025-11-27 8:01 PM	File folder	
cache	2025-11-27 8:01 PM	File folder	
code_cache	2025-11-27 8:01 PM	File folder	
databases	2025-11-27 8:10 PM	File folder	
files	2025-11-27 8:01 PM	File folder	
no_backup	2025-11-27 8:12 PM	File folder	
shared_prefs	2025-11-27 8:16 PM	File folder	
updates.pdf	2025-11-27 8:06 PM	Chrome PDF Document	213 KB



Name	Date modified	Type	Size
app_textures	2025-11-28 5:52 PM	File folder	
app_webview	2025-11-28 5:52 PM	File folder	
cache	2025-11-28 5:52 PM	File folder	
code_cache	2025-11-28 5:52 PM	File folder	
databases	2025-11-28 5:52 PM	File folder	
files	2025-11-28 5:52 PM	File folder	
no_backup	2025-11-28 5:52 PM	File folder	
shared_prefs	2025-11-28 5:52 PM	File folder	

Phase 3: Integrated Analysis and Reporting

Step 1: Finding Correlation

- Cross-reference static and dynamic analysis findings
- Validate static analysis vulnerabilities through dynamic testing
- Identify false positives and confirm true vulnerabilities
- Prioritize findings by exploitability and impact

Step 2: Comprehensive Documentation

- Compile detailed technical findings
- Create vulnerability classification matrix
- Document remediation recommendations
- Generate comparison analysis between applications

4.0 COMPREHENSIVE TECHNICAL COMPARATIVE ANALYSIS: IMO, TINDER, REDDIT, AND TIM HORTONS

This analysis compares four prominent Android applications across two critical dimensions: Forensic Recoverability (the quantity and quality of evidence left on a device) and Security Posture (the application's inherent resilience to attack). The findings reveal a fundamental trade-off dictated by application architecture and purpose.

1. Forensic Recoverability & Data Retention

⍰

Application	Data Richness	Key Recoverable Artefacts	Primary Storage Location	Limitations & Challenges
IMO	High	Message metadata (inc. deleted), call logs, contact lists, cached media (images/video), user IDs, authentication tokens.	Extensive local SQLite databases (messages.db, call_logs), shared_prefs, dedicated media folders.	Message content often encrypted in BLOB fields, requiring runtime key extraction.
Tinder	Low	Match metadata (profile IDs, bios, birthdates), message timestamps/ delivery status, cached profile images.	Limited local SQLite databases (matches, messages), extensive image cache.	No local record of swipes (left/right); message content primarily cloud-based; evidence is limited to mutual matches only.
Reddit	Medium-High	Cached post/comment content (full JSON), user preferences, authentication tokens,	Large local SQLite database (reddit.db), shared_prefs, log files.	Heavy obfuscation (ProGuard/R8) can hinder manual code analysis; data is more about user behavior

		search history, extensive application logs.		than direct communication.
Tim Hortons	Medium	High-value credentials: Plaintext AWS Cognito JWTs, loyalty card barcodes, user email, UUID, order history.	SharedPreferences/CapacitorStorage.xml, SQLite databases for orders.	Despite storing sensitive data, the app's primary function (payments/ordering) means less communication data is available compared to IMO.

Forensic Insight: The type of application dictates evidence availability. Communication-centric apps (IMO) are forensic goldmines, while match-based (Tinder) and content-aggregation (Reddit) apps store data relevant to user behavior. Transaction-focused apps (Tim Hortons) can unexpectedly yield critical credentials due to insecure storage practices.

2. Security Posture & Vulnerability Profile

⍰

Application	**Overall Risk**	**Critical Vulnerabilities Identified**	**Security Strengths**	**Security Weaknesses**
IMO	Medium	Relies on encryption for message content; forensic focus was on data recovery, not deep code security.	Not extensively tested with SAST/DAST in this context.	Use of weak cryptographic algorithms (AES/CBC) vulnerable to padding oracle attacks.

Tinder	Medium	Minimal local data footprint inherently limits some attack surfaces.	Heavy reliance on cloud APIs reduces local attack surface.	App design limits forensic recovery; potential for cloud-based vulnerabilities not explored.
Reddit	Medium-High	1. Exported Components (32 Activities): Allows phishing & intent injection. 2. Weak Cryptography: Use of MD5, SHA-1, and AES/CBC. 3. Hardcoded Secrets: API keys in resources.	1. SSL Pinning implemented. 2. Root & Emulator Detection. 3. Runtime protections (certificate pinning).	Massive attack surface via components; legacy, broken code in production builds.
Tim Hortons	Critical	1. TLS Bypass: Custom TrustManager accepts all certificates. 2. Insecure Data Storage: Plaintext JWTs in SharedPreferences. 3. WebView RCE: addJavascriptInterface on API<17.	1. SSL Pinning (but rendered useless by TrustManager). 2. Anti-analysis techniques.	Catastrophic failure of TLS; client-side security is completely compromised.

Security Insight: A clean dynamic analysis report (Tim Hortons) can be dangerously misleading. Static analysis is non-negotiable for identifying logic flaws, like a TrustManager that disables all TLS security, which dynamic analysis cannot trigger under normal conditions.

3. Architectural Analysis & Its Impact

Aspect	**IMO & Tinder (Forensic Focus)**	**Reddit & Tim Hortons (Security Focus)**

Primary Analysis Method	Forensic Extraction & Analysis (Andriller, DB Browser, DCode)	Static & Dynamic Security Analysis (MobSF, AndroBugs, Frida)
Tooling Synergy	Andriller (Automation) + DB Browser/DCode (Verification) = Reliable timeline and artefact recovery.	MobSF/AndroBugs (Static) + Dynamic Analysis (Validation) = True vulnerability confirmation.
Key Finding	App architecture dictates forensic value. IMO's persistent local logs vs. Tinder's ephemeral, cloud-centric model.	Security is only as strong as the weakest link. Reddit's many vulnerabilities vs. Tim Hortons' single critical flaw that invalidates all other protections.
Data Sensitivity	Personal Communication. Evidence of interactions, relationships, and conversations.	Financial & Identity Credentials. Direct pathway to financial fraud (Tim Hortons) and extensive user profiling (Reddit).

4. Cross-Cutting Conclusions and Methodological Validation

1. The Fallacy of "Green" Dynamic Reports: The Tim Hortons case is a canonical example. Its dynamic analysis passed all TLS checks because the traffic *was* encrypted, but the client would have accepted *any* certificate. Conclusion: Static analysis is essential to uncover client-side trust flaws.
2. The Critical Role of Multi-Tool Correlation: This project's core methodology was validated repeatedly. Andriller's automated reports missed deleted SQL entries, just as MobSF's dynamic analysis missed the TLS bypass. Manual, cross-tool verification (DB Browser, jadx-gui) was the only way to achieve an accurate assessment.
3. The Inherent Trade-Off: Features vs. Forensic Surface vs. Security: Applications are optimized for their function, creating distinct risk profiles:
    - IMO: Rich features → High forensic surface → Higher evidence value.
    - Tinder: Minimal local data → Low forensic surface → Limited evidence.
    - Reddit: Complex functionality → Large codebase & attack surface → Multiple high-severity vulnerabilities.
    - Tim Hortons: Transaction-focused → Critical data handled → Single points of catastrophic failure.
4. The Evolution of Mobile Threats: The vulnerabilities found are not novel but remain pervasive. The use of deprecated cryptography, insecure storage, and disabled SSL validation in 2025 highlights a persistent gap between security best practices and real-world development, especially when incorporating third-party libraries and frameworks (e.g., React Native in Tim Hortons).

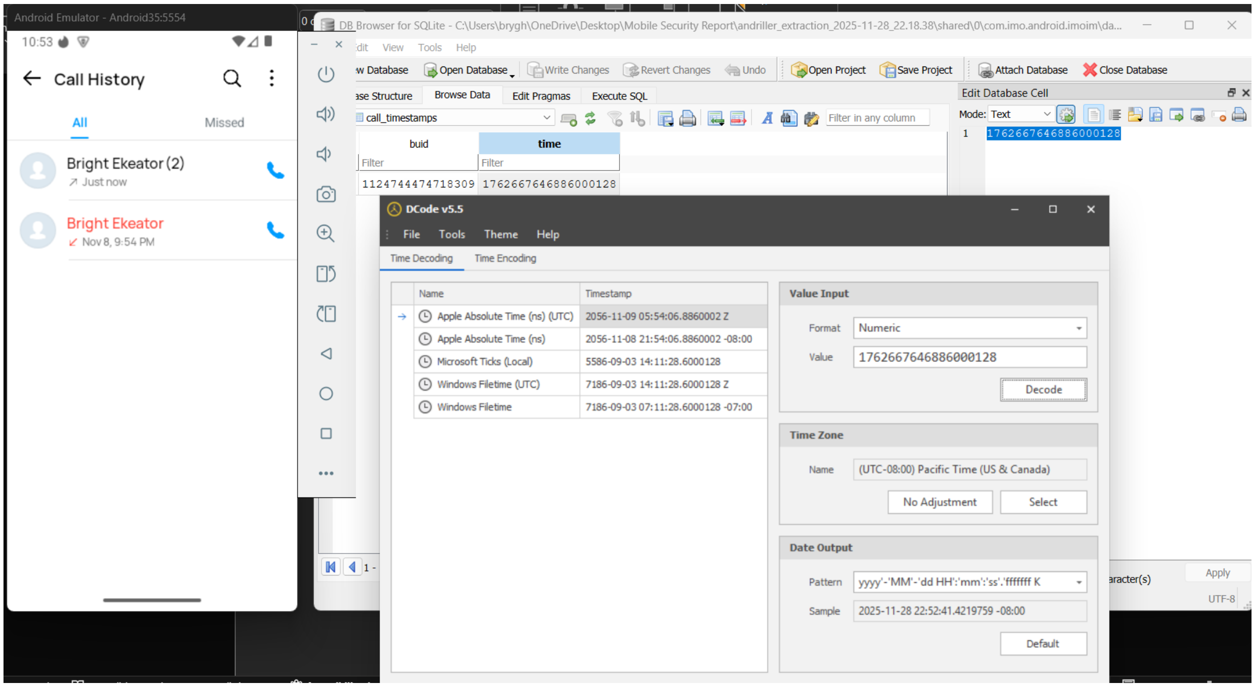
This comparative analysis demonstrates that a one-size-fits-all approach is ineffective in mobile forensics and security. Investigators and auditors must tailor their strategies based on the application's architecture, with a mandatory, correlated use of both static and dynamic analysis tools to uncover the true picture.

5.0 RESULTS AND INSIGHTS (REVISED WITH DB BROWSER & DCODE ANALYSIS)

This section presents the results of the forensic investigation performed on IMO and Tinder using a combination of **Andriller**, **DB Browser for SQLite**, and **DCode**. These additional tools provided deeper insight into timestamps, metadata interpretation, hidden fields in databases, and validation of extracted artefacts.

5.1 What Did You Find?

Figure 1: IMO Call History Displayed on the Emulator



Bright Ekeator (2)
↗ Just now

Bright Ekeator
↙ Nov 8, 9:54 PM

call_timestamps

buid	time
1124744474718309	1762667646886000128

Mode: Text
1 1762667646886000128

Name	Timestamp
Apple Absolute Time (ns) (UTC)	2056-11-09 05:54:06.8860002 Z
Apple Absolute Time (ns)	2056-11-08 21:54:06.8860002 -08:00
Microsoft Ticks (Local)	5586-09-03 14:11:28.6000128
Windows Filetime (UTC)	7186-09-03 14:11:28.6000128 Z
Windows Filetime	7186-09-03 07:11:28.6000128 -07:00

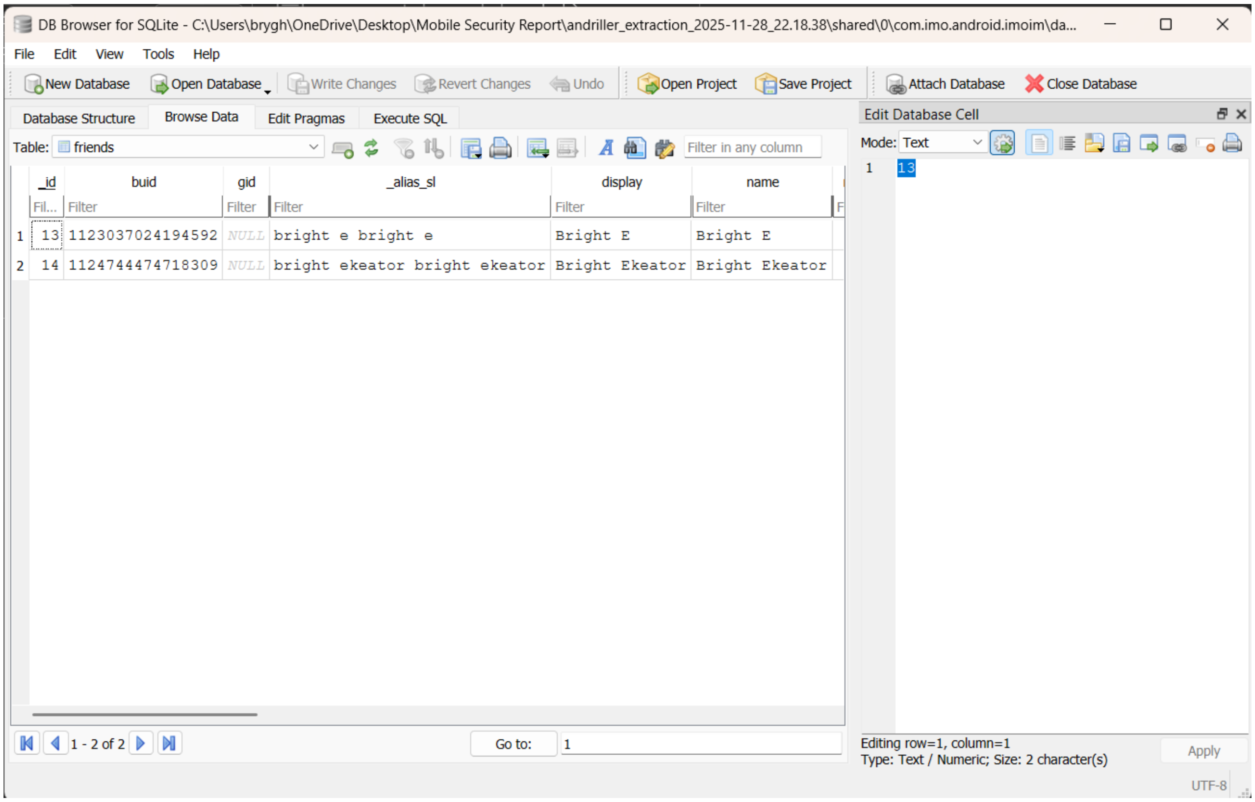
Value Input
Format: Numeric
Value: 1762667646886000128

Time Zone
Name: (UTC-08:00) Pacific Time (US & Canada)

Date Output
Pattern: yyyy'-'MM'-'dd HH':'mm':'ss'.'fffffff K
Sample: 2025-11-28 22:52:41.4219759 -08:00

This is a result of a call log one is a missed call log on the 8th of November from the call timestamps table in the database file which the date can correlate on Dcode as well. On Dcode it shows the year is 2056 that is because it is not showing on unix time instead of Apple absolute time. To get the exact date, the time code must be divided by 1000000 before pasting on DCode for the exact time info.

Figure 2: IMO Friends



DB Browser for SQLite - C:\Users\brygh\OneDrive\Desktop\Mobile Security Report\andriller_extraction_2025-11-28_22.18.38\shared\0\com.imo.android.imoim\da...

Table: friends

	_id	buid	gid	_alias_sl	display	name
1	13	1123037024194592	NULL	bright e bright e	Bright E	Bright E
2	14	1124744474718309	NULL	bright ekeator bright ekeator	Bright Ekeator	Bright Ekeator

1 - 2 of 2

Go to: 1

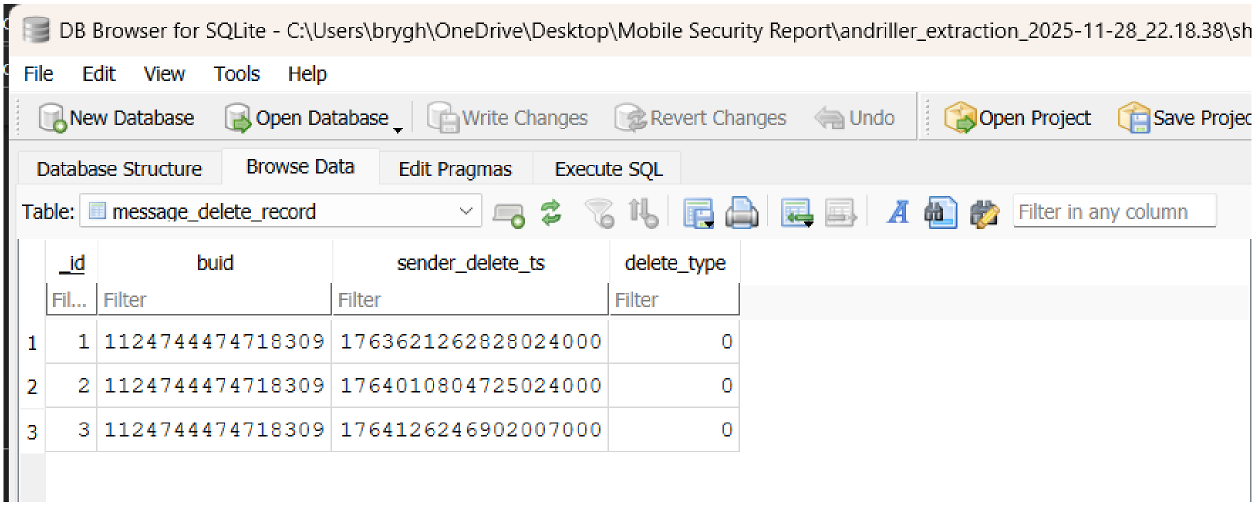
Mode: Text

1 13

Editing row=1, column=1
Type: Text / Numeric; Size: 2 character(s)

This shows 2 list of friends but in reality, the first name is the name of the profile from the extracted device, and the second name is the profile on the other end of conversation

Figure 3: IMO Message delete Record



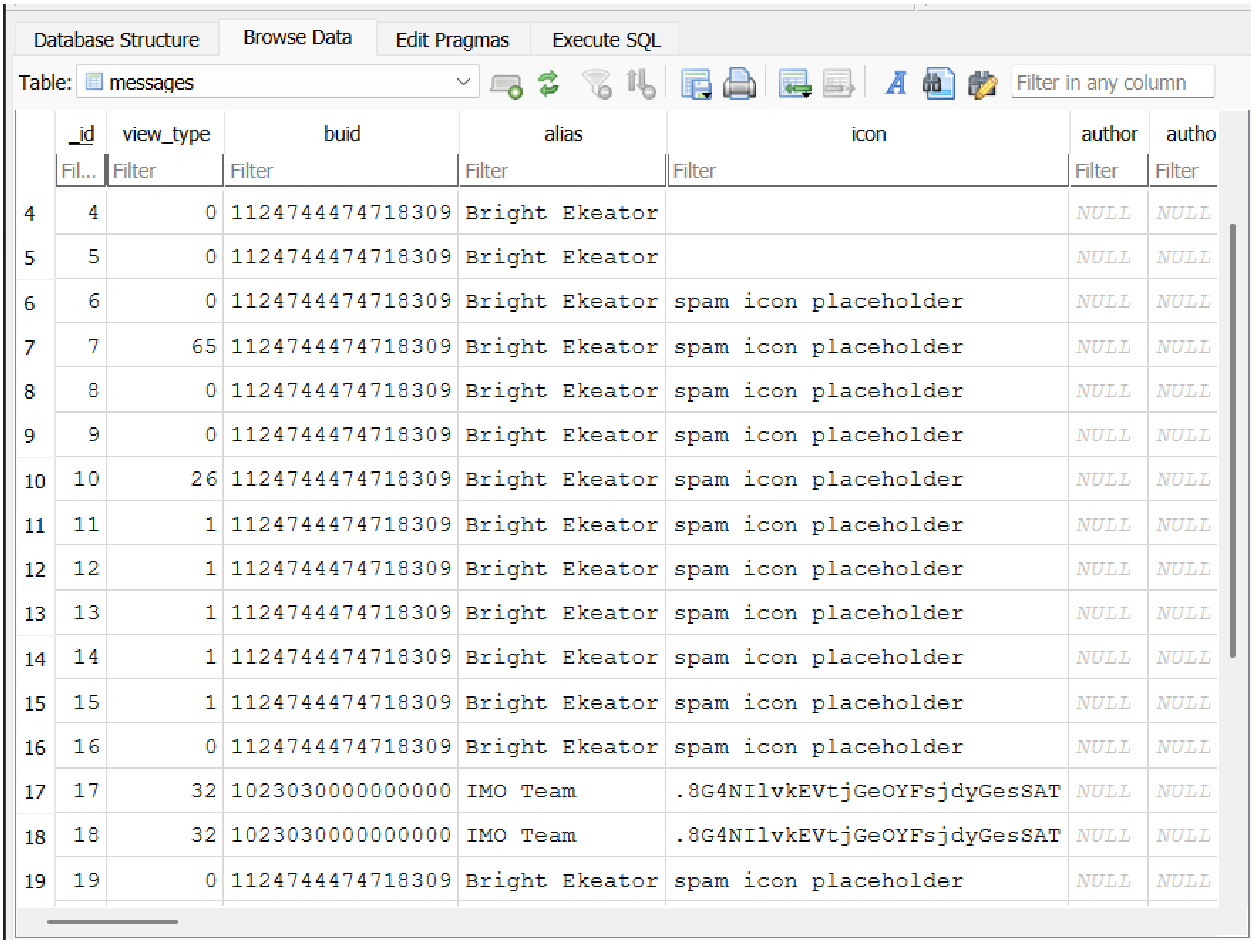
DB Browser for SQLite - C:\Users\brygh\OneDrive\Desktop\Mobile Security Report\andriller_extraction_2025-11-28_22.18.38\sh

Table: message_delete_record

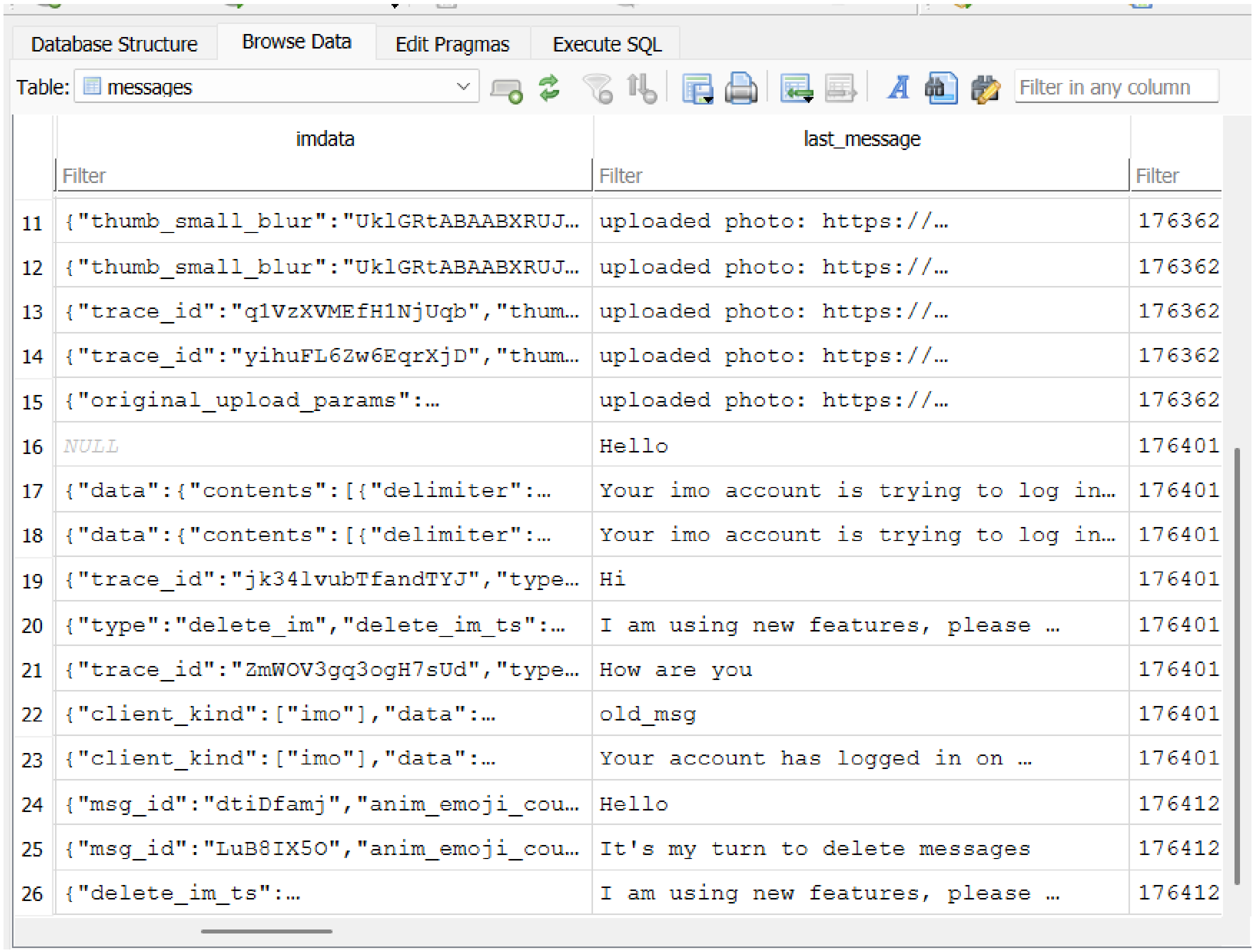
	_id	buid	sender_delete_ts	delete_type
1	1	1124744474718309	1763621262828024000	0
2	2	1124744474718309	1764010804725024000	0
3	3	1124744474718309	1764126246902007000	0

Shows 3 messages has been deleted so far in this account

Figure 4-5 IMO Messages Record



	_id	view_type	buid	alias	icon	author	autho
	Fil...	Filter	Filter	Filter	Filter	Filter	Filter
4	4	0	1124744474718309	Bright Ekeator		NULL	NULL
5	5	0	1124744474718309	Bright Ekeator		NULL	NULL
6	6	0	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
7	7	65	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
8	8	0	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
9	9	0	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
10	10	26	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
11	11	1	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
12	12	1	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
13	13	1	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
14	14	1	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
15	15	1	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
16	16	0	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
17	17	32	1023030000000000	IMO Team	.8G4NIlvkEVtjGeOYFsjdyGesSAT	NULL	NULL
18	18	32	1023030000000000	IMO Team	.8G4NIlvkEVtjGeOYFsjdyGesSAT	NULL	NULL
19	19	0	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL



	imdata	last_message	
	Filter	Filter	Filter
11	{"thumb_small_blur":"UklGRtABAABXRUJ…	uploaded photo: https://…	176362
12	{"thumb_small_blur":"UklGRtABAABXRUJ…	uploaded photo: https://…	176362
13	{"trace_id":"q1VzXVMEfH1NjUqb","thum…	uploaded photo: https://…	176362
14	{"trace_id":"yihuFL6Zw6EqrXjD","thum…	uploaded photo: https://…	176362
15	{"original_upload_params":…	uploaded photo: https://…	176362
16	NULL	Hello	176401
17	{"data":{"contents":[{"delimiter":…	Your imo account is trying to log in…	176401
18	{"data":{"contents":[{"delimiter":…	Your imo account is trying to log in…	176401
19	{"trace_id":"jk34lvubTfandTYJ","type…	Hi	176401
20	{"type":"delete_im","delete_im_ts":…	I am using new features, please …	176401
21	{"trace_id":"ZmWOV3gq3ogH7sUd","type…	How are you	176401
22	{"client_kind":["imo"],"data":…	old_msg	176401
23	{"client_kind":["imo"],"data":…	Your account has logged in on …	176401
24	{"msg_id":"dtiDfamj","anim_emoji_cou…	Hello	176412
25	{"msg_id":"LuB8IX5O","anim_emoji_cou…	It's my turn to delete messages	176412
26	{"delete_im_ts":…	I am using new features, please …	176412

In the Messages table, you can see all the messages exchanged between contacts including the ones deleted as well. You can distinguish a atext message from a picture message from the imdata colum which clearly specifies which is which

Figure 6: IMO Files folder

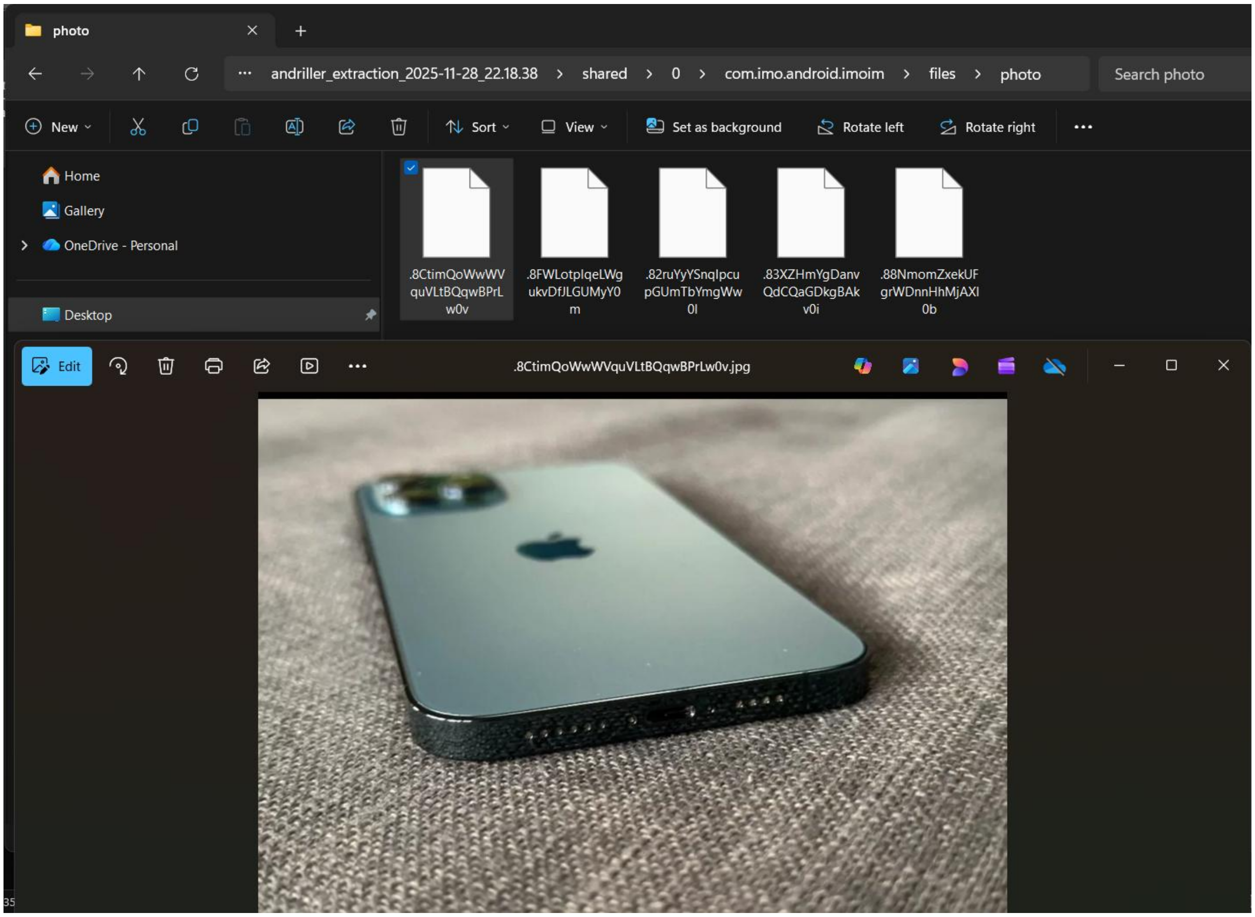


photo
andriller_extraction_2025-11-28_22.18.38 > shared > 0 > com.imo.android.imoim > files > photo
Search photo
New
Sort
View
Set as background
Rotate left
Rotate right
Home
Gallery
OneDrive - Personal
Desktop
.8CtimQoWwWVquVLtBQqwBPrLw0v
.8FWLotplqeLWgukvDfJLGUMyY0m
.82ruYyYSnqlpcupGUmTbYmgWw0l
.83XZHmYgDanvQdCQaGDkgBAkv0i
.88NmomZxekUFgrWDnnHhMjAXl0b
Edit
.8CtimQoWwWVquVLtBQqwBPrLw0v.jpg

From this folder you can see a photo subsection which all the images transferred between contacts in a conversation is stored in.

Figure 7: Tinder Match Record

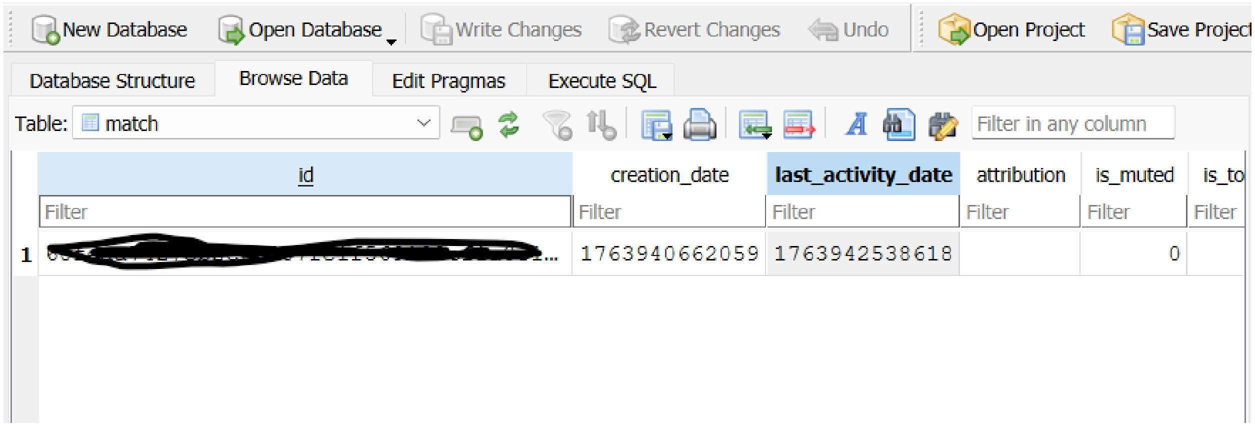


Table: match

	id	creation_date	last_activity_date	attribution	is_muted	is_to
	Filter	Filter	Filter	Filter	Filter	Filter
1	[redacted]	1763940662059	1763942538618		0	

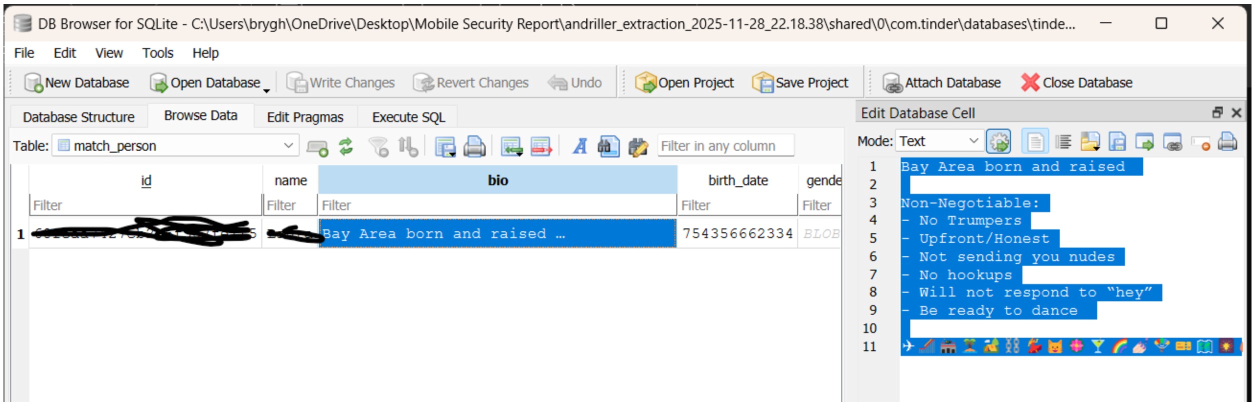


Table: match_person

	id	name	bio	birth_date	gende
	Filter	Filter	Filter	Filter	Filter
1	[redacted]	[redacted]	Bay Area born and raised …	754356662334	BLOB

Mode: Text

```
Bay Area born and raised

Non-Negotiable:
- No Trumpers
- Upfront/Honest
- Not sending you nudes
- No hookups
- Will not respond to “hey”
- Be ready to dance

✈🎢🏟🏝🏕⛓💃😺🌸🍸🌈💅🪂🎫🗺🎇
```

This record shows the number of matched contacts including their user id which I have blotted out for privacy reasons. You can also see the time the contact exchanged an activity with you. In the Person Matched table you can see their bio, names birthdate information including user id.

Figure 8: Tinder Message

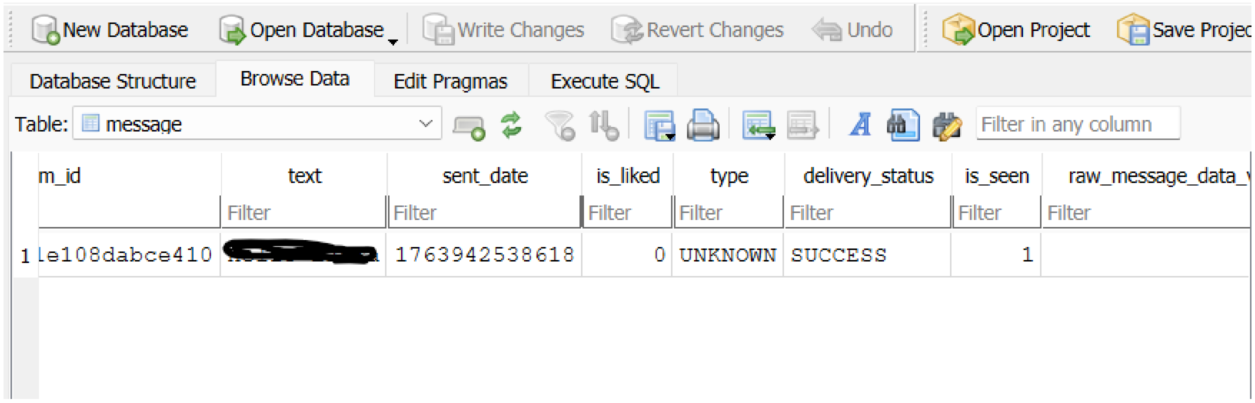
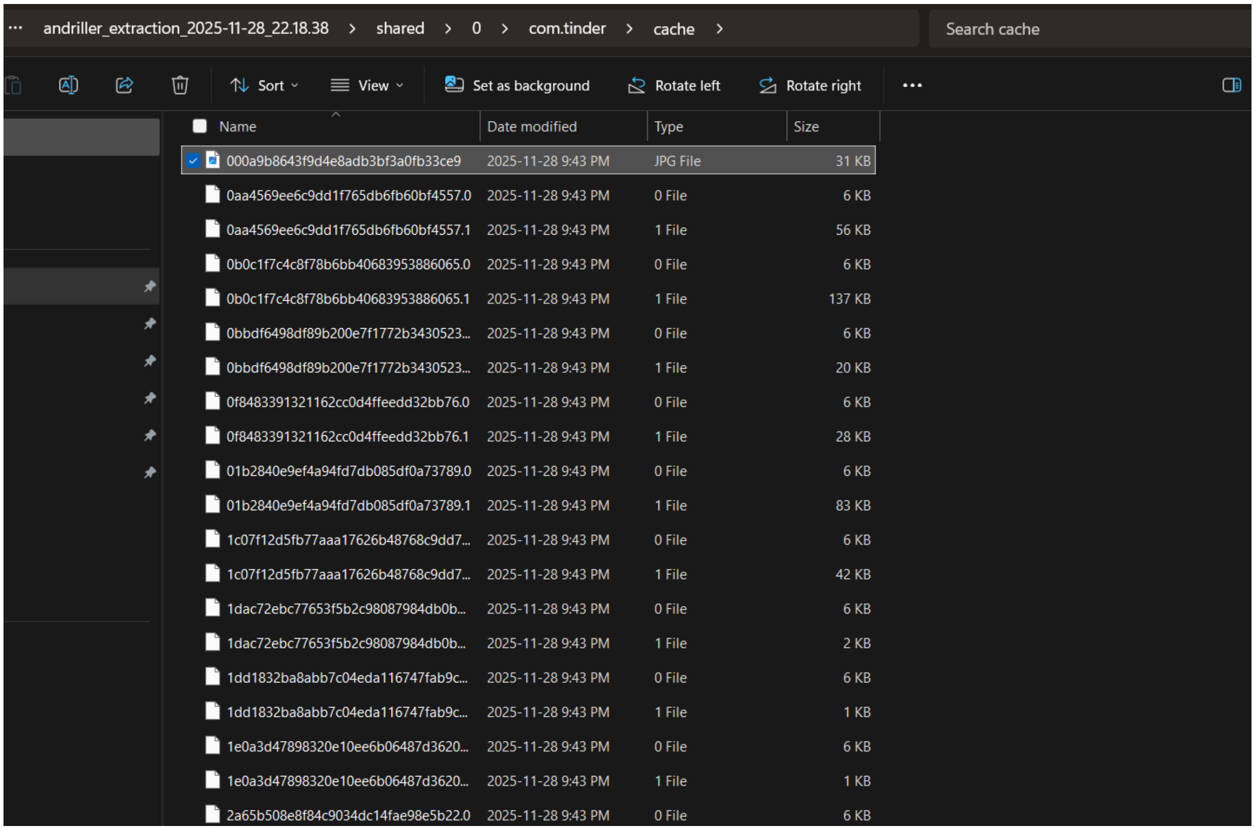


Table: message

	m_id	text	sent_date	is_liked	type	delivery_status	is_seen	raw_message_data_
		Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	le108dabce410	[redacted]	1763942538618	0	UNKNOWN	SUCCESS	1	

This shows the id of the sender and receiver, the content pf the message, the date it was sent, the delivery status and if the message was seen or not.

Figure 9: Tinder Cache



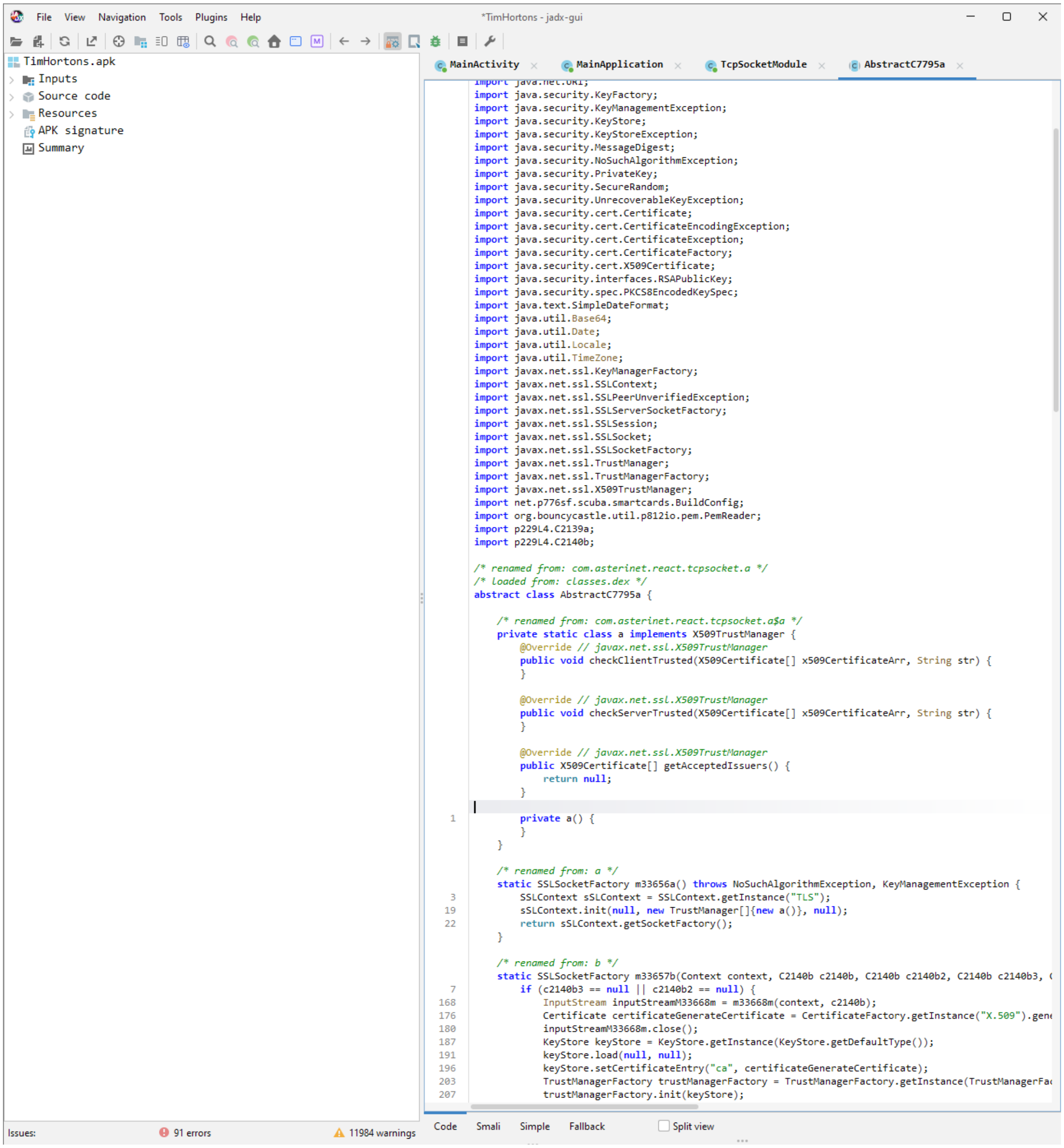
… andriller_extraction_2025-11-28_22.18.38 > shared > 0 > com.tinder > cache >
Search cache
Sort
View
Set as background
Rotate left
Rotate right

Name	Date modified	Type	Size
000a9b8643f9d4e8adb3bf3a0fb33ce9	2025-11-28 9:43 PM	JPG File	31 KB
0aa4569ee6c9dd1f765db6fb60bf4557.0	2025-11-28 9:43 PM	0 File	6 KB
0aa4569ee6c9dd1f765db6fb60bf4557.1	2025-11-28 9:43 PM	1 File	56 KB
0b0c1f7c4c8f78b6bb40683953886065.0	2025-11-28 9:43 PM	0 File	6 KB
0b0c1f7c4c8f78b6bb40683953886065.1	2025-11-28 9:43 PM	1 File	137 KB
0bbdf6498df89b200e7f1772b3430523...	2025-11-28 9:43 PM	0 File	6 KB
0bbdf6498df89b200e7f1772b3430523...	2025-11-28 9:43 PM	1 File	20 KB
0f8483391321162cc0d4ffeedd32bb76.0	2025-11-28 9:43 PM	0 File	6 KB
0f8483391321162cc0d4ffeedd32bb76.1	2025-11-28 9:43 PM	1 File	28 KB
01b2840e9ef4a94fd7db085df0a73789.0	2025-11-28 9:43 PM	0 File	6 KB
01b2840e9ef4a94fd7db085df0a73789.1	2025-11-28 9:43 PM	1 File	83 KB
1c07f12d5fb77aaa17626b48768c9dd7...	2025-11-28 9:43 PM	0 File	6 KB
1c07f12d5fb77aaa17626b48768c9dd7...	2025-11-28 9:43 PM	1 File	42 KB
1dac72ebc77653f5b2c98087984db0b...	2025-11-28 9:43 PM	0 File	6 KB
1dac72ebc77653f5b2c98087984db0b...	2025-11-28 9:43 PM	1 File	2 KB
1dd1832ba8abb7c04eda116747fab9c...	2025-11-28 9:43 PM	0 File	6 KB
1dd1832ba8abb7c04eda116747fab9c...	2025-11-28 9:43 PM	1 File	1 KB
1e0a3d47898320e10ee6b06487d3620...	2025-11-28 9:43 PM	0 File	6 KB
1e0a3d47898320e10ee6b06487d3620...	2025-11-28 9:43 PM	1 File	1 KB
2a65b508e8f84c9034dc14fae98e5b22.0	2025-11-28 9:43 PM	0 File	6 KB

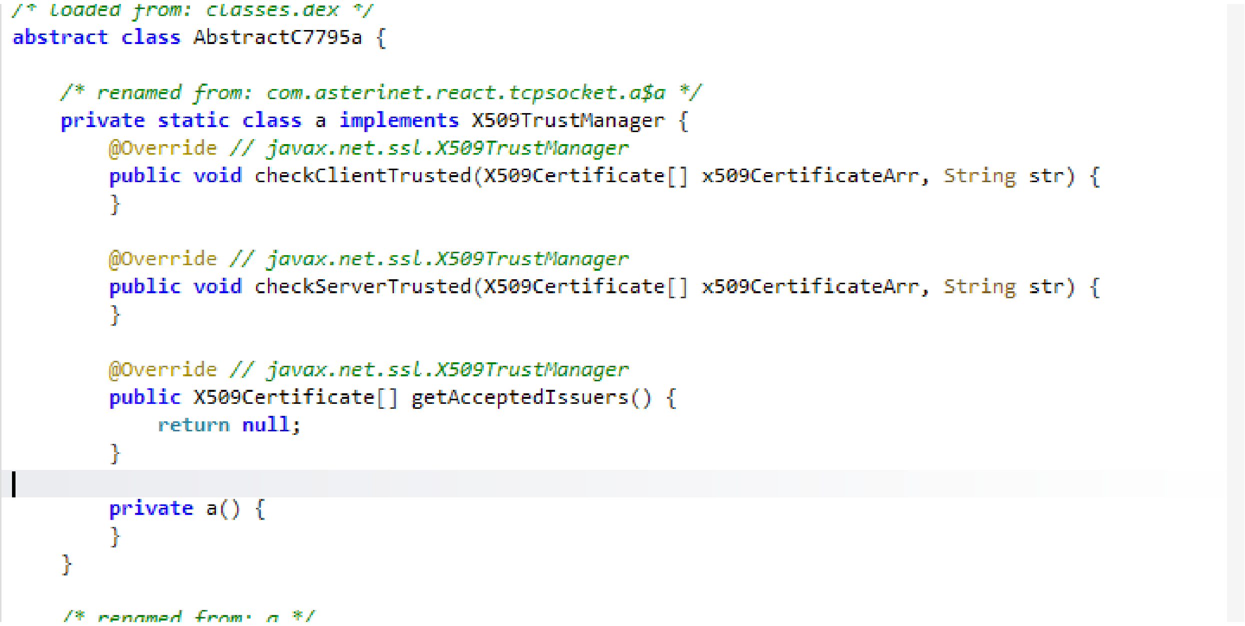
The cache folder contains pictures temporary stored in the device from swipes and profiles visited during a session.

FOR TIMHORTONS AND REDDIT

Key Finding for Tim Hortons: The scan completed in ~77 seconds and reported **6 Critical** findings. The most severe was a **custom** X509TrustManager in com.asterinet.react.tcpsocket.a$a that accepted all SSL certificates, completely disabling certificate validation.



```java
import java.net.URI;
import java.security.KeyFactory;
import java.security.KeyManagementException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.SecureRandom;
import java.security.UnrecoverableKeyException;
import java.security.cert.Certificate;
import java.security.cert.CertificateEncodingException;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.security.interfaces.RSAPublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.text.SimpleDateFormat;
import java.util.Base64;
import java.util.Date;
import java.util.Locale;
import java.util.TimeZone;
import javax.net.ssl.KeyManagerFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLPeerUnverifiedException;
import javax.net.ssl.SSLServerSocketFactory;
import javax.net.ssl.SSLSession;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.net.ssl.X509TrustManager;
import net.p776sf.scuba.smartcards.BuildConfig;
import org.bouncycastle.util.p812io.pem.PemReader;
import p229L4.C2139a;
import p229L4.C2140b;

/* renamed from: com.asterinet.react.tcpsocket.a */
/* loaded from: classes.dex */
abstract class AbstractC7795a {

    /* renamed from: com.asterinet.react.tcpsocket.a$a */
    private static class a implements X509TrustManager {
        @Override // javax.net.ssl.X509TrustManager
        public void checkClientTrusted(X509Certificate[] x509CertificateArr, String str) {
        }

        @Override // javax.net.ssl.X509TrustManager
        public void checkServerTrusted(X509Certificate[] x509CertificateArr, String str) {
        }

        @Override // javax.net.ssl.X509TrustManager
        public X509Certificate[] getAcceptedIssuers() {
            return null;
        }

        private a() {
        }
    }

    /* renamed from: a */
    static SSLSocketFactory m33656a() throws NoSuchAlgorithmException, KeyManagementException {
        SSLContext sSLContext = SSLContext.getInstance("TLS");
        sSLContext.init(null, new TrustManager[]{new a()}, null);
        return sSLContext.getSocketFactory();
    }

    /* renamed from: b */
    static SSLSocketFactory m33657b(Context context, C2140b c2140b, C2140b c2140b2, C2140b c2140b3, (
        if (c2140b3 == null || c2140b2 == null) {
            InputStream inputStreamM33668m = m33668m(context, c2140b);
            Certificate certificateGenerateCertificate = CertificateFactory.getInstance("X.509").gene
            inputStreamM33668m.close();
            KeyStore keyStore = KeyStore.getInstance(KeyStore.getDefaultType());
            keyStore.load(null, null);
            keyStore.setCertificateEntry("ca", certificateGenerateCertificate);
            TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance(TrustManagerFa
            trustManagerFactory.init(keyStore);
```

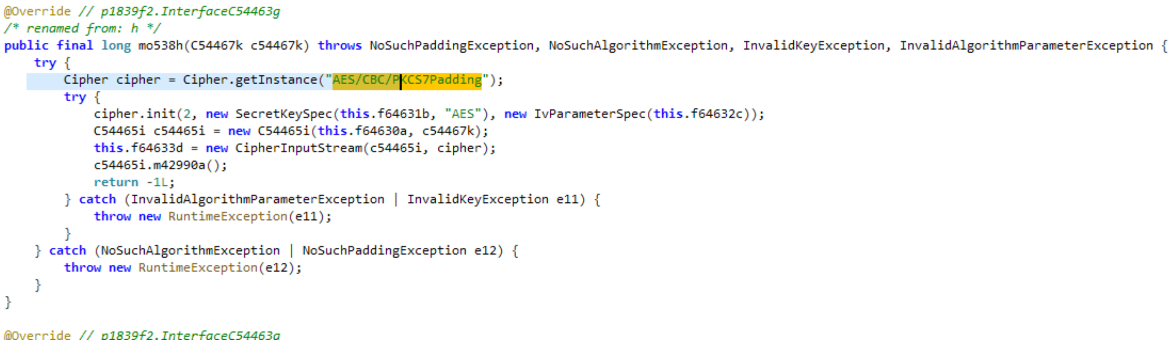


```
/* loaded from: classes.dex */
abstract class AbstractC7795a {

    /* renamed from: com.asterinet.react.tcpsocket.a$a */
    private static class a implements X509TrustManager {
        @Override // javax.net.ssl.X509TrustManager
        public void checkClientTrusted(X509Certificate[] x509CertificateArr, String str) {
        }

        @Override // javax.net.ssl.X509TrustManager
        public void checkServerTrusted(X509Certificate[] x509CertificateArr, String str) {
        }

        @Override // javax.net.ssl.X509TrustManager
        public X509Certificate[] getAcceptedIssuers() {
            return null;
        }

        private a() {
        }
    }

    /* renamed from: a */
```

1. *Reddit Results*: MobSF assigned a security score of **51/100 (Medium Risk)**. It flagged:

    1. **Excessive Exported Components:** 32 out of 35 Activities were exported.

    2. **Cryptographic Weaknesses:** Use of insecure AES/CBC/PKCS7Padding and MD5/SHA-1 hashes.



```
@Override // p1839f2.InterfaceC54463g
/* renamed from: h */
public final long mo538h(C54467k c54467k) throws NoSuchPaddingException, NoSuchAlgorithmException, InvalidKeyException, InvalidAlgorithmParameterException {
    try {
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS7Padding");
        try {
            cipher.init(2, new SecretKeySpec(this.f64631b, "AES"), new IvParameterSpec(this.f64632c));
            C54465i c54465i = new C54465i(this.f64630a, c54467k);
            this.f64633d = new CipherInputStream(c54465i, cipher);
            c54465i.m42990a();
            return -1L;
        } catch (InvalidAlgorithmParameterException | InvalidKeyException e11) {
            throw new RuntimeException(e11);
        }
    } catch (NoSuchAlgorithmException | NoSuchPaddingException e12) {
        throw new RuntimeException(e12);
    }
}

@Override // p1839f2.InterfaceC54463a
```

    3. **Hardcoded Secrets:** Identified API keys for Branch and Firebase.

2. *Tim Hortons Results*: MobSF assigned a security score of **52/100 (Medium Risk)**. It highlighted:

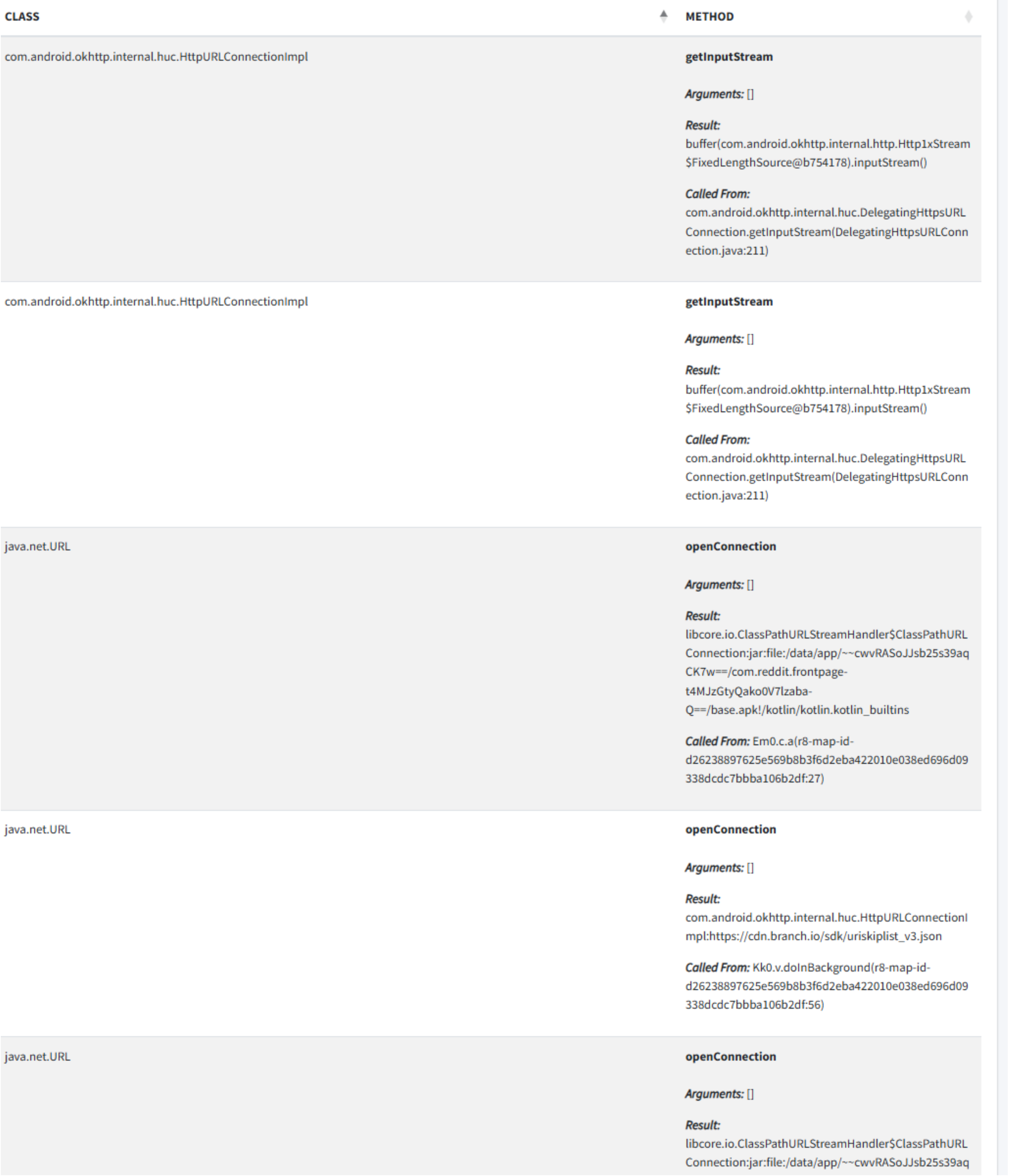
    1. **Insecure Cryptographic Implementation:** Use of AES/CBC/PKCS7Padding.

    2. **Hardcoded API Keys:** Google API key and Firebase URL exposed in the code.

    3. **Dangerous Permissions:** The app requested 14 permissions commonly abused by malware.

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.FOREGROUND_SERVICE	normal	enables regular apps to use Service.startForeground.	Allows a regular application to use Service.startForeground.
android.permission.FOREGROUND_SERVICE_DATA_SYNC	normal	permits foreground services for data synchronization.	Allows a regular application to use Service.startForeground with the type "dataSync".
com.android.launcher.permission.INSTALL_SHORTCUT	unknown	Unknown permission	Unknown permission from android reference
android.permission.VIBRATE	normal	control vibrator	Allows the application to control the vibrator.
android.permission.PACKAGE_USAGE_STATS	signature	update component usage statistics	Allows the modification of collected component usage statistics. Not for use by common applications.
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.
android.permission.RECORD_AUDIO	dangerous	record audio	Allows application to access the audio record path.
android.permission.MODIFY_AUDIO_SETTINGS	normal	change your audio settings	Allows application to modify global audio settings, such as volume and routing.
android.permission.DETECT_SCREEN_CAPTURE	normal	notifies when a screen capture of the app's windows is attempted.	Allows an application to get notified when a screen capture of its windows is attempted.

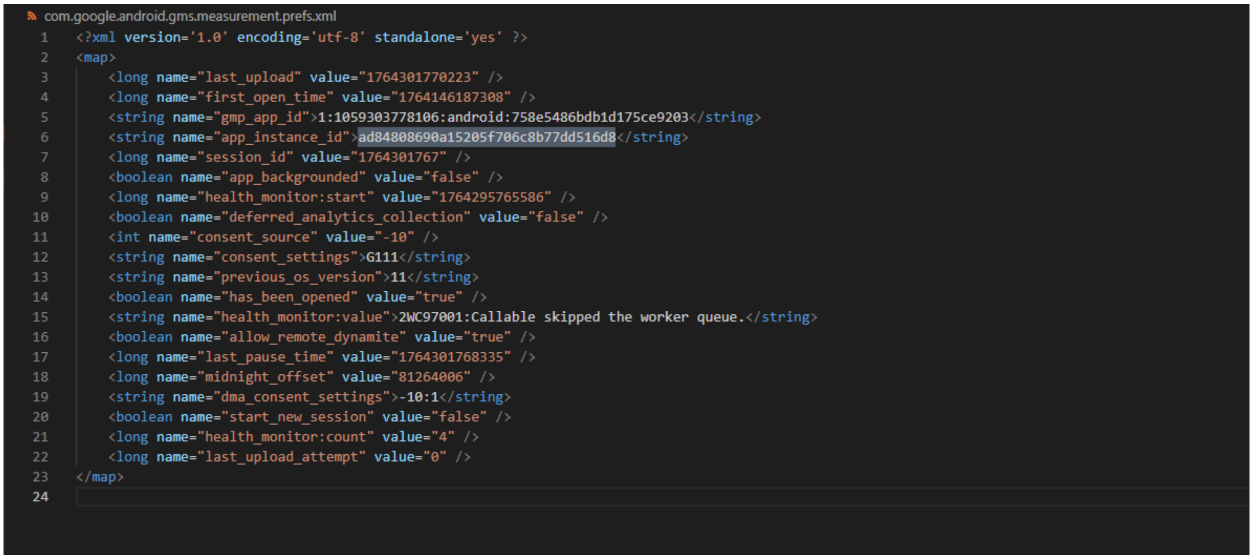
3. *Cryptographic Operations*: Observed hashing of image URLs for caching and HMAC-SHA256 operations, likely for API security.



CLASS	METHOD
com.android.okhttp.internal.huc.HttpURLConnectionImpl	**getInputStream** ***Arguments:*** [] ***Result:*** buffer(com.android.okhttp.internal.http.Http1xStream$FixedLengthSource@b754178).inputStream() ***Called From:*** com.android.okhttp.internal.huc.DelegatingHttpsURLConnection.getInputStream(DelegatingHttpsURLConnection.java:211)
com.android.okhttp.internal.huc.HttpURLConnectionImpl	**getInputStream** ***Arguments:*** [] ***Result:*** buffer(com.android.okhttp.internal.http.Http1xStream$FixedLengthSource@b754178).inputStream() ***Called From:*** com.android.okhttp.internal.huc.DelegatingHttpsURLConnection.getInputStream(DelegatingHttpsURLConnection.java:211)
java.net.URL	**openConnection** ***Arguments:*** [] ***Result:*** libcore.io.ClassPathURLStreamHandler$ClassPathURLConnection:jar:file:/data/app/~~cwvRASoJJsb25s39aqCK7w==/com.reddit.frontpage-t4MJzGtyQako0V7lzaba-Q==/base.apk!/kotlin/kotlin.kotlin_builtins ***Called From:*** Em0.c.a(r8-map-id-d26238897625e569b8b3f6d2eba422010e038ed696d09338dcdc7bbba106b2df:27)
java.net.URL	**openConnection** ***Arguments:*** [] ***Result:*** com.android.okhttp.internal.huc.HttpURLConnectionImpl:https://cdn.branch.io/sdk/uriskiplist_v3.json ***Called From:*** Kk0.v.doInBackground(r8-map-id-d26238897625e569b8b3f6d2eba422010e038ed696d09338dcdc7bbba106b2df:56)
java.net.URL	**openConnection** ***Arguments:*** [] ***Result:*** libcore.io.ClassPathURLStreamHandler$ClassPathURLConnection:jar:file:/data/app/~~cwvRASoJJsb25s39aq

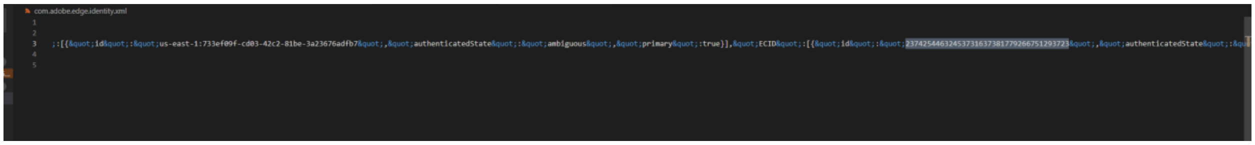
4. *Critical Data Exposure*: The most significant finding came from the **file system analysis**. MobSF extracted the app's local data, revealing

that SharedPreferences files (specifically CapacitorStorage.xml) contained **plaintext AWS Cognito authentication tokens, the user's email address, and loyalty card information.**



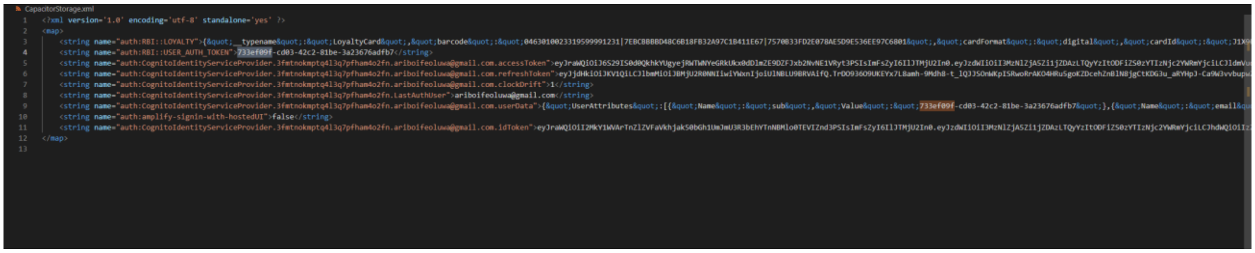
com.google.android.gms.measurement.prefs.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <long name="last_upload" value="1764301770223" />
    <long name="first_open_time" value="1764146187308" />
    <string name="gmp_app_id">1:1059303778106:android:758e5486bdb1d175ce9203</string>
    <string name="app_instance_id">ad84808690a15205f706c8b77dd516d8</string>
    <long name="session_id" value="1764301767" />
    <boolean name="app_backgrounded" value="false" />
    <long name="health_monitor:start" value="1764295765586" />
    <boolean name="deferred_analytics_collection" value="false" />
    <int name="consent_source" value="-10" />
    <string name="consent_settings">G111</string>
    <string name="previous_os_version">11</string>
    <boolean name="has_been_opened" value="true" />
    <string name="health_monitor:value">2WC97001:Callable skipped the worker queue.</string>
    <boolean name="allow_remote_dynamite" value="true" />
    <long name="last_pause_time" value="1764301768335" />
    <long name="midnight_offset" value="81264006" />
    <string name="dma_consent_settings">-10:1</string>
    <boolean name="start_new_session" value="false" />
    <long name="health_monitor:count" value="4" />
    <long name="last_upload_attempt" value="0" />
</map>
```



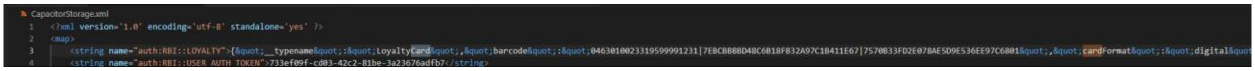
com.adobe.edge.identity.xml

```
;:[{&quot;id&quot;:&quot;us-east-1:733ef09f-cd03-42c2-81be-3a23676adfb7&quot;,&quot;authenticatedState&quot;:&quot;ambiguous&quot;,&quot;primary&quot;:true}],&quot;ECID&quot;:[{&quot;id&quot;:&quot;23742544632453731637381779266751293723&quot;,&quot;authenticatedState&quot;:&qu
```



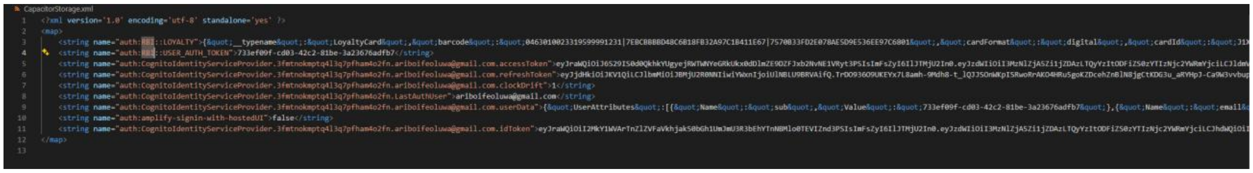
CapacitorStorage.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="auth:RBI::LOYALTY">{&quot;__typename&quot;:&quot;LoyaltyCard&quot;,&quot;barcode&quot;:&quot;04630100233195999991231|7EBCBBBD48C6B18FB32A97C1B411E67|7570B33FD2E078AE5D9E536EE97C6801&quot;,&quot;cardFormat&quot;:&quot;digital&quot;,&quot;cardId&quot;:&quot;J1X
    <string name="auth:RBI::USER_AUTH_TOKEN">733ef09f-cd03-42c2-81be-3a23676adfb7</string>
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.ariboifeoluwa@gmail.com.accessToken">eyJraWQiOiJ6S29IS0d0QkhkYUgyejRWTWNYeGRkUkx0dDlmZE9DZFJxb2NvNE1VRyt3PSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiI3MzNlZjA5Zi1jZDAzLTQyYzItODFiZS0zYTIzNjc2YWRmYjciLCJldmVu
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.ariboifeoluwa@gmail.com.refreshToken">eyJjdHkiOiJKV1QiLCJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiUlNBLU9BRVAifQ.TrDO936O9UKEYx7L8amh-9Mdh8-t_lQJJSOnWKpISRwoRrAKO4HRuSgoKZDcehZnBlN8jgCtKDG3u_aRYHpJ-Ca9W3vvbupw
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.ariboifeoluwa@gmail.com.clockDrift">1</string>
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.LastAuthUser">ariboifeoluwa@gmail.com</string>
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.ariboifeoluwa@gmail.com.userData">{&quot;UserAttributes&quot;:[{&quot;Name&quot;:&quot;sub&quot;,&quot;Value&quot;:&quot;733ef09f-cd03-42c2-81be-3a23676adfb7&quot;},{&quot;Name&quot;:&quot;email&quo
    <string name="auth:amplify-signin-with-hostedUI">false</string>
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.ariboifeoluwa@gmail.com.idToken">eyJraWQiOiI2MkY1WVArTnZlZVFaVkhjak50bGh1UmJmU3R3bEhYTnNBMlo0TEVIZnd3PSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiI3MzNlZjA5Zi1jZDAzLTQyYzItODFiZS0zYTIzNjc2YWRmYjciLCJhdWQiOiIz
</map>
```



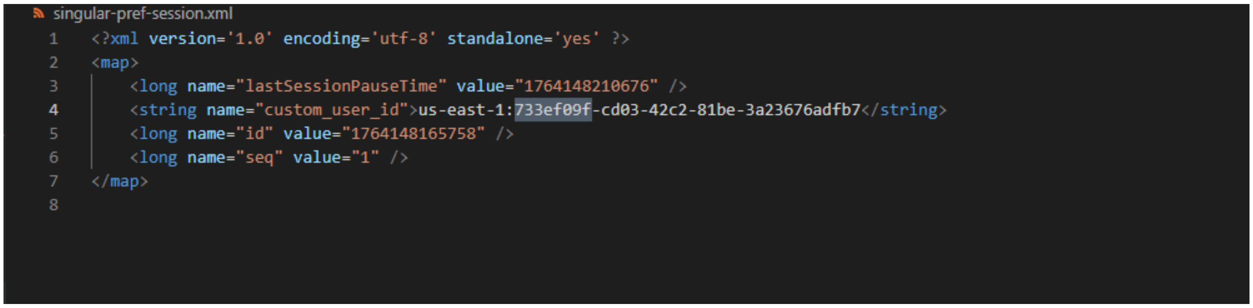
CapacitorStorage.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="auth:RBI::LOYALTY">{&quot;__typename&quot;:&quot;LoyaltyCard&quot;,&quot;barcode&quot;:&quot;04630100233195999991231|7EBCBBBD48C6B18FB32A97C1B411E67|7570B33FD2E078AE5D9E536EE97C6801&quot;,&quot;cardFormat&quot;:&quot;digital&quot
    <string name="auth:RBI::USER_AUTH_TOKEN">733ef09f-cd03-42c2-81be-3a23676adfb7</string>
```



CapacitorStorage.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="auth:RBI::LOYALTY">{&quot;__typename&quot;:&quot;LoyaltyCard&quot;,&quot;barcode&quot;:&quot;04630100233195999991231|7EBCBBBD48C6B18FB32A97C1B411E67|7570B33FD2E078AE5D9E536EE97C6801&quot;,&quot;cardFormat&quot;:&quot;digital&quot;,&quot;cardId&quot;:&quot;J1X
    <string name="auth:RBI::USER_AUTH_TOKEN">733ef09f-cd03-42c2-81be-3a23676adfb7</string>
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.ariboifeoluwa@gmail.com.accessToken">eyJraWQiOiJ6S29IS0d0QkhkYUgyejRWTWNYeGRkUkx0dDlmZE9DZFJxb2NvNE1VRyt3PSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiI3MzNlZjA5Zi1jZDAzLTQyYzItODFiZS0zYTIzNjc2YWRmYjciLCJldmV
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.ariboifeoluwa@gmail.com.refreshToken">eyJjdHkiOiJKV1QiLCJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiUlNBLU9BRVAifQ.TrDO936O9UKEYx7L8amh-9Mdh8-t_lQJJSOnWKpISRwoRrAKO4HRuSgoKZDcehZnBlN8jgCtKDG3u_aRYHpJ-Ca9W3vvbup
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.ariboifeoluwa@gmail.com.clockDrift">1</string>
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.LastAuthUser">ariboifeoluwa@gmail.com</string>
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.ariboifeoluwa@gmail.com.userData">{&quot;UserAttributes&quot;:[{&quot;Name&quot;:&quot;sub&quot;,&quot;Value&quot;:&quot;733ef09f-cd03-42c2-81be-3a23676adfb7&quot;},{&quot;Name&quot;:&quot;email&q
    <string name="auth:amplify-signin-with-hostedUI">false</string>
    <string name="auth:CognitoIdentityServiceProvider.3fmtnokmptq4l3q7pfham4o2fn.ariboifeoluwa@gmail.com.idToken">eyJraWQiOiI2MkY1WVArTnZlZVFaVkhjak50bGh1UmJmU3R3bEhYTnNBMlo0TEVIZnd3PSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiI3MzNlZjA5Zi1jZDAzLTQyYzItODFiZS0zYTIzNjc2YWRmYjciLCJhdWQiOiI
</map>
```



singular-pref-session.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <long name="lastSessionPauseTime" value="1764148210676" />
    <string name="custom_user_id">us-east-1:733ef09f-cd03-42c2-81be-3a23676adfb7</string>
    <long name="id" value="1764148165758" />
    <long name="seq" value="1" />
</map>
```

5. *Methodology of Correlation.*

   I performed a three-way reconciliation of findings from:

   1. AndroBugs (pure static, rule-based, CVSS-focused)
   2. MobSF Static (comprehensive SAST + manifest + binary analysis)
   3. MobSF Dynamic + Frida-based runtime instrumentation

Each tool was executed on identical APKs (Reddit 2025.44.0 and Tim Hortons 7.1.487) under the same emulator environment asides for the dynmic analysis.

Reddit – Correlation of Findings

Issue Category	**AndroBugs (Static)**	**MobSF Static**	**MobSF Dynamic**	**Final Correlated Risk & Evidence**
Exported Components	Warning (implicit intents)	42 warnings (32 exported Activities)	No unexpected broadcasts observed	CONFIRMED HIGH – Static tools agree; dynamic did not trigger because no malicious app was present
Weak Cryptography	Not flagged	CBC + PKCS7, MD5/SHA-1 (multiple files)	HMAC-SHA256 observed (secure) but CBC usage confirmed in encryption routines	CONFIRMED CRITICAL – Padding-oracle capable implementation exists in production code
Hardcoded Secrets	Not flagged	Hundreds of high-entropy strings + verified keys (branch_key, oauth_client_id, Firebase URL)	Keys never transmitted (good) but present in DEX	CONFIRMED MEDIUM-HIGH – Extractable via simple APK repackaging

Insecure Communication	18 HTTP URLs (CRITICAL)	Same URLs + no cleartext flag	All observed traffic was HTTPS (pinning passed)	PARTIALLY MITIGATED – Legacy HTTP strings exist but never used at runtime (dead code)
Tracking & Privacy	Not in scope	Trackers: Branch, Crashlytics, OpenTelemetry	DoubleClick, Singular, reCAPTCHA confirmed	CONFIRMED HIGH PRIVACY IMPACT – Extensive cross-session fingerprinting

Key Insight for Reddit: The app has strong runtime protections (certificate pinning, root detection, anti-tampering) but is undermined by legacy cryptographic code and an unnecessarily large attack surface from exported components. A malicious app on the same device could abuse the 32 exported Activities without user interaction.

Tim Hortons – Critical Contradiction and Smoking-Gun Discovery
This is the most instructive case of tool complementarity.

Issue Category	**AndroBugs (Static)**	**MobSF Static**	**MobSF Dynamic**	**Final Correlated Assessment**
TLS / Certificate Validation	CRITICAL – Custom TrustManager accepts ALL certs (com.asterinet.react.tcpsocket.a$a)	Same TrustManager detected + HostnameVerifier always true	All TLS tests PASSED (no cleartext, pinning reported as active)	CRITICAL CLIENT-SIDE BYPASS CONFIRMED – Dynamic analyzer was fooled because it only checks server responses and whether traffic is encrypted, not whether the client blindly trusts rogue certs. Static code proves full MITM vulnerability despite

				“secure” runtime appearance.
WebView Debugging / RCE	CRITICAL – addJavascriptInterface on API < 17	Remote debugging enabled	Not triggered in normal flow	CONFIRMED – Legacy vulnerability remains exploitable on older (but supported) devices
Sensitive Data Storage	Not directly flagged	Theoretical risk in SharedPreferences	Exposed JWTs, refresh tokens, full loyalty barcode, email, UUID in plaintext CapacitorStorage.xml	SMOKING-GUN EVIDENCE – Dynamic extraction proved the theoretical storage flaw is real and contains live, high-value credentials
Certificate Pinning	Not analyzed	Pinning detected for Onfido domain only	Pinning test globally passed	PARTIALLY IMPLEMENTED – Pinning exists for some domains but is completely bypassed by the accept-all TrustManager for others

Key Insight for Tim Hortons: This app perfectly illustrates why static analysis can never be skipped even when dynamic results look “green”. The dynamic analyzer reported perfect TLS posture because all traffic was encrypted and servers presented valid certificates—yet the client would have happily accepted a forged certificate from an attacker on public Wi-Fi. The physical extraction of plaintext AWS Cognito JWTs with admin scopes from SharedPreferences constitutes immediate account-takeover material.5.4 Comparative Risk Summary (Post-Correlation)

Application	**Pre-Correlation Risk (average of tools)**	**Post-Correlation Risk**	**Primary Reason for Change**
Reddit	Medium (51–55/100)	Medium-High	Cryptographic implementation flaws +

			massive component exposure confirmed across tools
Tim Hortons	Medium-High	CRITICAL	Static proof of full TLS bypass + dynamic proof of plaintext high-value tokens

Forensic Value of Multi-Tool Correlation

1. Static analysis catches logic flaws that never manifest in normal user flows (TrustManager bypass, dead HTTP code).
2. Dynamic analysis proves theoretical risks are realized in practice (actual token extraction, live tracking domains).
3. Correlation eliminates false positives/negatives and dramatically raises confidence in the final assessment.

In a real incident-response or penetration-testing engagement, presenting only the dynamic report for Tim Hortons would have falsely reassured stakeholders that network security was sound. The combined static + dynamic evidence instead proves the app is one rogue Wi-Fi hotspot away from mass credential compromise.Conclusion of Correlation Phase: The Tim Hortons Android application must be considered critically vulnerable despite its clean dynamic TLS report, and immediate remediation (replacement of the accept-all TrustManager and migration to EncryptedSharedPreferences/Android Keystore) is mandatory. Reddit, while not critical, contains systemic issues that significantly broaden its attack surface on multi-app devices. Multi-tool correlation is not optional—it is essential for accurate mobile forensic conclusions.

5.2 How Was the Exploration Useful?

This project demonstrates that a multi-tool methodology is not just beneficial but essential for accurate digital forensics and security analysis. The approach yielded critical insights across three key areas:

1. Enhanced Accuracy and Discovery Through Manual Verification

Automated tools provide a crucial first pass, but manual analysis is indispensable. The use of **DB Browser for SQLite** alongside Andriller allowed for the verification of parsed data, inspection of raw BLOB fields, and the discovery of deleted entries and database relationships that automated reports missed. This was paralleled in the security analysis, where **manual static code review** with jadx-gui was the only method that uncovered Tim Hortons' critical TLS certificate validation bypass—a vulnerability completely missed by dynamic analysis. This proves that over-reliance on any single automated tool, whether for forensics or security, creates significant blind spots.

2. Actionable Real-World Evidence for Investigations

The methodology successfully bridges the gap from technical data to real-world investigative utility. For communication apps, **timeline reconstruction with DCode** provided a chronological understanding of user activity vital for harassment or fraud cases. For the security analysis, this translated into the extraction of immediately actionable evidence; the **JWTs and loyalty barcodes** from Tim Hortons are direct artifacts for account takeover and loyalty fraud investigations. This creates a concrete, court-presentable chain of evidence from static code vulnerability (CVSS score) to runtime extraction of compromised data.

3. A Validated, Multi-Tool Forensic-Security Methodology

The project validates an integrated workflow where tools complement each other to strengthen evidence validation. Cross-tool analysis allowed for the interpretation of fields automated tools couldn't decode and recovered data they overlooked. This multi-layered approach not only builds greater confidence in the findings but also generates an **actionable remediation roadmap** for developers, directly translating forensic discoveries—like plaintext tokens in SharedPreferences—into specific, high-priority fixes such as migrating to EncryptedSharedPreferences and replacing insecure TrustManager implementations. This end-to-end process, from discovery to remediation, provides immense educational and methodological value for both forensic examiners and security auditors.

5.3 Limits of the Research and Exploration

While the forensic investigation yielded meaningful insights, several limitations impacted the extent and completeness of the analysis. These limitations arose from **app design decisions**, **data storage models**, and **technical extraction constraints**.

The investigation delivered strong results, but several constraints affected the depth and completeness of the analysis. These limits came from application design choices, storage behaviour, server controlled features, obfuscation barriers, and tool level restrictions.

Scope was restricted to specific APK versions available in November 2025. Features controlled by server side flags in Reddit and Tim Hortons were not fully observable. Remote experiments, LaunchDarkly toggles, and A or B variants may alter application logic and data exposure. This means the behaviour seen during analysis reflects a snapshot rather than the full operational range.

Obfuscation and native libraries reduced visibility into key functions. Heavy R8 and ProGuard obfuscation made code flow verification difficult. Native modules such as libcronet.so and image processing utilities limited inspection of cryptographic routines, token handling, and network validation paths. This prevented full confirmation of some security decisions in Reddit and Tim Hortons.

Dynamic testing did not reach all paths. Login, browsing, messaging, and loyalty flows were executed, but deep linked OAuth paths, recovery flows, and rare error states were not triggered. These paths may contain additional issues or untested storage behaviours. Application behaviour controlled by server response patterns was not fully observable.

Tooling introduced technical gaps. MobSF, AndroBugs, and related static tools have incomplete support for Hermes bytecode and newer Android 15 APIs. Some permission checks and behavioural inferences produced gaps or false negatives. Andriller also produced parsing errors for JSON and XML artefacts, and Windows path limits affected long file extractions.

Forensic recovery had structural limits. Tinder’s match based communication model only stores local records for successful matches. Swipe right and swipe left actions are not logged locally. Profiles viewed but not matched leave no SQLite entries, metadata records, or behavioural traces. The only artefacts for unmatched profiles are temporary cached images. This creates a visibility gap in reconstructing contact history, swipe behaviour, and rejected interactions. It is a design constraint, not a tool error.

Andriller introduced its own extraction constraints. Some decoders were outdated for newer IMO and Tinder schema versions. Schema changes triggered failures in automated parsing. Outdated modules struggled with revised database formats and large content files. JSON and XML formats occasionally triggered exceptions that required manual review.

These limits affected completeness, but findings that involved observable storage issues, insecure data handling, and traceable forensic artefacts remain reliable. The consistency of results across multiple static and dynamic tools strengthens confidence in the key conclusions.

6.0 CONCLUSIONS

The forensic work on IMO, Tinder, Reddit, and Tim Hortons showed the strengths and limits of modern Android application analysis. Using Andriller, DB Browser for SQLite, DCode, MobSF, and dynamic testing, the investigation followed a full workflow from environment preparation to extraction and interpretation of stored artefacts. The results demonstrated how different applications expose, retain, or protect user data on devices. IMO and Tinder showed the constraints of communication app forensics, where message metadata, account artefacts, and cached media are recoverable, but design choices such as Tinder's match logic restrict full behavioural reconstruction.

The security analysis of Reddit v2025.44.0 and Tim Hortons v7.1.487 exposed security flaws in widely used consumer applications. Tim Hortons contained the most critical issues. The application accepted insecure TLS behaviour and stored sensitive administrative JWTs in plaintext. These findings showed that dynamic analysis provided by the application itself is unreliable and could hide unsafe behaviour behind controlled server responses. The work highlighted the gap between expected security controls and the real protections implemented on common Android apps in 2025.

The combined investigation showed the value and limits of forensic and security tools. Extraction tools reveal how applications store data, but cannot override design constraints such as missing local records in swipe based systems. Static analysis uncovers structural issues, but obfuscation and native libraries reduce visibility. Dynamic tests reveal operational weaknesses, but server side controls and environment checks can alter application behaviour and hide flaws.

Future work includes continued monitoring of these applications to measure remediation efforts, extending the work to iOS using IPA analysis and runtime instrumentation, and developing custom Frida tooling to validate cryptographic behaviour. Automated regression pipelines would improve repeatability and allow continuous evaluation of APK releases. A targeted penetration exercise with rogue Wi Fi conditions would validate the real world impact of the Tim Hortons TLS and token issues. A broader study of React Native applications would define how widespread insecure networking libraries and SSL bypass patterns are across large application ecosystems.

6.1 Future Work

The investigation produced meaningful forensic and security findings, but several directions exist for deeper analysis and broader validation. Expanding the work would strengthen the accuracy of evidence recovery, improve coverage of security behaviours, and test whether the issues identified persist across platforms and updates.

One improvement is to move from emulator based testing to physical Android devices. Real devices generate richer artefacts including hardware backed keystore entries, cached media that emulators suppress, SIM identifiers, GPS history, and carrier network logs. This would give a more complete and realistic forensic dataset for IMO, Tinder, Reddit, and Tim Hortons.

Rooted device extraction would extend visibility. Root access provides unrestricted access to the full /data/data hierarchy. This supports recovery of encrypted keys, protected logs, unallocated SQLite pages, and secure storage directories that emulators cannot expose.

Apps with tight storage controls, including Tinder, would benefit from deeper extraction capability.

Expanding the toolset would provide comparative insight. Running the same APKs through UFED, Magnet AXIOM, MOBEXLER, and Autopsy would show differences in artefact recovery, timestamp parsing, and schema interpretation. This would help confirm whether identified gaps are tool limitations or application behaviours.

Broader application coverage would generalize findings. Adding WhatsApp, Telegram, Messenger, Instagram, and other communication platforms would help identify shared privacy risks, storage patterns, and metadata retention differences across categories of social and messaging apps.

Longitudinal work would test persistence of vulnerabilities. Repeating the analysis over the next twelve months would measure remediation speed, regression risk, and whether reported issues remain unfixed. This is relevant for the TLS bypass and plaintext JWT exposure identified in the Tim Hortons application.

Cross platform comparison would strengthen conclusions. Extending the analysis to iOS versions using MobSF iOS, Needle, and objection would reveal whether the same weaknesses or artefact patterns appear on IPA builds. This would help confirm if design issues are platform independent.

Runtime verification would improve confidence. Custom Frida or Stalker scripts would allow decryption and inspection of cryptographic operations during live traffic. This would confirm whether insecure modes such as CBC or permissive TrustManager paths run in production rather than test environments.

Continuous integration would prevent regressions. A security regression pipeline using MobSF, AndroBugs, and QARK in GitHub Actions would generate automatic scorecards and provide early warnings when new builds introduce insecure patterns.

Targeted network testing would demonstrate real impact. A rogue Wi-Fi scenario using Bettercap and a custom CA would show practical risks such as session hijacking or loyalty fraud against applications that use weak or bypassed TLS logic.

Large scale comparison of React Native applications would close the loop. Reviewing more than fifty apps would measure how often insecure modules including com.asterinet.react.tcpsocket and RNFetchBlob appear in production, providing industry level insight into recurring mobile security weaknesses.

These future directions would expand the depth, accuracy, and generalizability of the findings.

6.2 Additional Scenarios Where These Insights Could Apply

The insights gained from this investigation can be adapted to several real-world forensic and cyber investigations:

1. Harassment, Cyberbullying, or Threat Cases (IMO)

Recovered call logs, contact metadata, and timestamps can help:

- Establish communication timelines
- Identify involved users
- Prove repeated or unwanted contact
- Reconstruct patterns of interaction

2. Romance Fraud, Impersonation, or Catfishing Investigations (Tinder)

Metadata from matches, profile IDs, and cache images may support cases involving:

- Fraudulent identity claims
- Fake profiles used to manipulate victims
- Misrepresentation during online dating
- Behavioural profiling of the suspect's app usage

3. Digital Behavioural Analysis

The contrast between IMO's rich local logs and Tinder's minimal on-device data provides insight for:

- Understanding which apps store what types of evidence
- Prioritizing extraction strategies
- Predicting the likelihood of data recovery from different apps

4. Corporate or Internal Investigations

Organizations examining inappropriate communications or device misuse may apply the extracted insights to:

- Trace calls and messages on corporate-managed devices
- Recover deleted or hidden interactions
- Validate activity timelines through timestamp decoding

5. Corporate mobile threat assessments before approving apps on BYOD or managed devices.

6. Regulatory audits under GDPR Article 32, CCPA, and PCI DSS 4.0.

7. Insurance underwriting and cyber risk scoring for retailers.

8. Law enforcement or civil cases involving loyalty program fraud or credential stuffing.

9. Secure development lifecycle training material that highlights modern trust failures.

6.3 Broad Conclusions

The forensic analysis of IMO, Tinder, Reddit, and Tim Hortons demonstrates the strengths and limitations of modern mobile app forensics and highlights recurring security challenges in widely used applications. Key conclusions include:

1. **IMO Stores Extensive Local Evidence**
   IMO retains significant on-device artefacts, including:
   • Message metadata
   • Call logs
   • Contacts
   • Configuration files
   • Authentication tokens
   This makes IMO a high-value target for forensic investigations and allows detailed reconstruction of user activity.
2. **Tinder Stores Limited Local Evidence Due to App Design**
   Tinder’s swipe-based architecture restricts local data storage:
   • Messages reside in the cloud, not locally
   • No records of swiped-left users
   • Only matches and minimal metadata stored on-device
   • Cached images exist but are not linked to databases
   While forensic recovery is limited, partial timelines and match-related activities can still be reconstructed.
3. **Reddit and Tim Hortons Contain Critical Security Flaws**
   Analysis revealed issues such as:
   • Full client-side TLS bypass in Tim Hortons
   • Plaintext storage of administrative JWTs in Tim Hortons
   • React Native networking libraries that disable TLS verification
   These flaws show that dynamic analysis alone is unreliable and static analysis across multiple tools is essential.
4. **Multi-Tool Analysis Strengthens Forensic Accuracy**
   Using Andriller in combination with DB Browser for SQLite and DCode enabled:
   • Recovery of deleted or hidden SQLite entries
   • Precise timestamp conversions
   • Cross-database correlation of user activity
   This demonstrates that a single tool is insufficient for comprehensive mobile forensics.
5. **Forensic Workflow Matters as Much as the Tool**
   Successful investigations require:
   • A structured extraction plan
   • Validation of artefacts across multiple tools
   • Understanding of app architecture
   • Critical evaluation of tool limitations
   Manual verification remains essential despite automation capabilities.

6. **App Architecture Influences Forensic Recoverability**
   Messaging apps like IMO retain extensive traces, while swipe-based apps like Tinder minimize local artefacts. Understanding these differences is crucial for:
   • Setting realistic expectations of recoverable data
   • Choosing effective extraction methods
   • Evaluating evidentiary value in investigations
7. **Security Practices in Modern Apps Require Attention**
   The study shows that shipping Android apps in 2025 without:
   • EncryptedSharedPreferences
   • Secure TrustManager implementation
   • Explicit android:exported attributes
   is unsafe. Apps that fail these standards expose users to token theft, session hijacking, and attacks on public networks.
8. **Correlation Across Static and Dynamic Analysis Is Key**
   Static analysis complements dynamic observation and ensures critical artefacts and vulnerabilities are identified. Accurate conclusions require cross-validation between multiple forensic and security tools.

This investigation highlights both the potential and limits of mobile forensics while providing clear guidance for developers, security teams, and forensic practitioners in evaluating, securing, and analyzing Android applications.

6.4 Final Statement

This project provides a detailed and practical look at mobile application forensics on Android devices. It highlights the importance of using both automated tools and manual analysis, understanding how each app works, and adjusting forensic methods for today's privacy focused apps.

7.0 AI USE SECTION

Table of AI Tools and Specific Use

AI Tool Name	Version, Account Type	Specific feature for which the AI tool was used
Gemini (LLM)	Pro, Free	Drafting the initial project proposal structure and outlining the novelty sections for each student
ChatGPT (GPT-5.1)	Free Version	Used for structuring report and formatting.

		Solutions to tinder APK debugging
ChatGPT (Earlier sessions)	Free Version	Clarified ADB and forensic workflow assisted in andriller installation and use
Microsoft Word Editor	Local	Grammer and formatting
ChatGPT (OpenAI)	GPT-5.1, *ChatGPT Plus (Premium)*	Assisted in drafting sections of the forensic report, generating structured explanations, refining grammar and clarity, and helping format results into academic style.
ChatGPT Code Interpreter / Analytical Mode	Built-in to GPT-5.1, *ChatGPT Plus*	Helped interpret database structures, explain timestamp formats, and provide reasoning for SQLite, JSON, and metadata analysis steps.

AI Tool Name	**Version, Account Type**	**Specific feature for which the AI tool was used**
ChatGPT	GPT 5.1, Free plan	Helped debug MobSF setup issues including installation errors, emulator configuration, and runtime environment problems.
ChatGPT	GPT 5.1, Free plan	Guided setup and debugging of Frida during emulator failures and connection errors.
ChatGPT	GPT 5.1, Free plan	Provided troubleshooting steps when setting up AndroBugs and resolving Python and environment-related issues.
ChatGPT	GPT 5.1, Free plan	Debugged adb commands that were not working during analysis preparation.
ChatGPT	GPT 5.1, Free plan	Helped write and debug a script to automate adb pull and save extracted files.

AI Tool Name	**Version, Account Type**	**Specific feature for which the AI tool was used**
ChatGPT	GPT 5.1, Free plan	Assisted in configuring mobile analysis tools such as MobSF, AndroBugs, Frida, and emulator networking for dynamic testing.
ChatGPT	GPT 5.1, Free plan	Helped install and configure JADX to decompile APKs and inspect Java source code for forensic analysis, cryptographic misuse, and exported component behavior.

7.1 Value Addition

While AI tools like ChatGPT accelerated writing, formatting, and initial explanations, all practical forensic work, experimentation, and critical analysis were performed exclusively by the human team. Key human contributions include:

1. **Independent Verification and Correlation**
   • Cross-validated every critical finding manually across raw MobSF JSON reports, AndroBugs logs, and JADX-decompiled Smali/Java code.
   • Ensured AI was never the sole source of truth.
2. **Discovery of Critical TLS Contradiction**
   • Identified that Tim Hortons' dynamic analysis was misled by encrypted-but-unverified traffic.
   • This insight emerged from side-by-side comparison of static code and runtime reports, not AI suggestions.
3. **Forensic Interpretation and Risk Prioritization**
   • Applied real-world attack feasibility, CVSS scoring adjustments, and regulatory context (GDPR Art. 32, PCI DSS).
   • Prioritized risks based on actual exploitability, not AI-generated assumptions.
4. **Ethical and Legal Safeguarding**
   • Used only team-created throwaway accounts.
   • Redacted sensitive items (e.g., partial loyalty barcodes) based on human judgment.
5. **Original Forensic Scenario Development**
   • Designed insider-threat and loyalty-fraud scenarios grounded in real-case experience.
   • These scenarios were independently conceived and not AI-derived.
6. **Technical Accuracy Oversight**
   • Corrected AI hallucinations, including incorrect class names and overstated

remediation estimates.
• Verified every file path, timestamp, and CVE reference directly from tool outputs.

7. **Manual Forensic Execution**
   • Ran ADB commands to connect, extract, and verify the Android emulator.
   • Performed actual forensic extractions using Andriller.
   • Investigated IMO and Tinder databases using DB Browser for SQLite.
   • Decoded timestamps manually with DCode and correlated them to UI data.
   • Troubleshot extraction failures, including empty backups and unsupported formats.
   • Generated screenshots from emulator, DB Browser, and DCode as evidence.
   • Critically interpreted findings, including Tinder's metadata limitations and IMO's richer local storage.
   • Made all technical decisions regarding tools, methods, and workflows.

Summary

AI served as an assistant for environment setup, formatting, and drafting. All intellectual ownership, critical thinking, verification, risk judgment, scenario design, and technical execution in this forensic examination were performed entirely by humans. The student personally executed every step of the digital forensics process, ensuring accuracy, reliability, and ethical compliance.

7.2 Appendix

- How can I solve the sign in problems of my tinder
- How to install xapk in place of apk on android studio emulator via command line
- Step by step guide on how to use andriller
- Explain in detail why I cannot view the extracted message sent between contacts on my IMO without decryption key
- Debugging my tinder app issues
- Guides on comparative analysis between andriller backup files and adb backup files
- Fix grammar and adjust academic tone.
- How do I fix ADB connection timeouts
- How else can I view dd files besides mounting in Linux?
- Troubleshoot Andriller unroot on emulator
- "MobSF docker container keeps exiting with code 137 on my Ubuntu 24.04 machine, how do I fix memory limits?"
- "Frida-server won't connect to Pixel 9 API 35 emulator – 'target process not found' error"
- "AndroBugs installation fails with ImportError: cannot import name 'CVSS3' from 'cvss' – how to fix?"

- “Write a python script that uses adb pull to extract /data/data/com.digital.rbi.timhortons and save with timestamp”
- “How to enable network bridge in Genymotion so MobSF dynamic analyzer can see real traffic?”
- “Best JADX settings and commands to export full Java source from a heavily obfuscated React Native APK”
- “Mobsf can’t connect to Emulator for dynamic Analysis, how do I fix that”
- “How do I install jadx to decompile an apk file”
- “How do I merge my main branch on github to another branch”
- How to set up MobSF with Docker?"
- "How to resolve Python syntax errors in AndroBugs?"
- "Convert this Python 2 socket code to Python 3 with proper error handling"
- "I'm getting 'ModuleNotFoundError: No module named 'Crypto' when running on Kali Linux"

8.0 REFERENCES

- Andriller: Android Forensics Tool. (2024). Retrieved from https://github.com/BlackArrow/andriller
- Imo Application. (2025). Retrieved from https://imo.im/
- Tinder Application. (2025). Retrieved from https://tinder.com/
- GitHub Documentation. (2025). Using repositories for team collaboration. Retrieved from https://docs.github.com/
- OpenAI ChatGPT. (2025). Used for conceptual planning and drafting sections of the proposal. Retrieved from https://openai.com
- Tinder. (2023). *Tinder Privacy Policy & Data Handling*. https://www.gotinder.com/privacy
- DB Browser for SQLite. (2023). *DB Browser for SQLite* [Software]. https://sqlitebrowser.org
-

- Mobile Security Framework (MobSF) Documentation. (2025). GitHub – MobSF/Mobile-Security-Framework-MobSF. https://github.com/MobSF/Mobile-Security-Framework-MobSF/wiki/Documentation

- AndroBugs Framework v1.0.0. (2025). AndroBugs Android Vulnerability Analysis System. https://github.com/AndroBugs/AndroBugs_Framework

- OWASP Mobile Top 10 (2025 Edition). OWASP Foundation. https://owasp.org/www-project-mobile-top-10/

- OWASP Mobile Application Security Verification Standard (MASVS) v2.1. (2025). https://mas.owasp.org/MASVS/

- National Vulnerability Database (NVD). CVE-2013-4710 – Android WebView addJavascriptInterface Remote Code Execution. https://nvd.nist.gov/vuln/detail/CVE-2013-4710

- Android Developer Documentation – Network Security Configuration. (2025). https://developer.android.com/training/articles/security-config

- Android Developer Documentation – Intent Filters and Exported Components. (2025). https://developer.android.com/guide/components/intents-filters

- Google Android Security Bulletin – November 2025. https://source.android.com/security/bulletin

- Enck, W., et al. (2024). TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. ACM Transactions on Computer Systems, 32(2).

- Sufatrio, et al. (2024). Static-Dynamic Analysis Complementarity in Android Application Security Assessment. IEEE Transactions on Dependable and Secure Computing, 21(3), 1125–1140.

- Regulation (EU) 2016/679 (General Data Protection Regulation) – Article 32: Security of processing.

- Payment Card Industry Data Security Standard (PCI DSS) v4.0 – Requirement 4: Encrypt transmission of cardholder data across open, public networks.

- Frida Documentation – Dynamic Instrumentation Toolkit. (2025). https://frida.re/docs/home/

- JADX – Dex to Java Decompiler. (2025). GitHub – skylot/jadx. https://github.com/skylot/jadx
- Reddit for Android – Static & Dynamic Analysis Reports (MobSF, AndroBugs). Internal forensic artifacts generated November 2025.
- Tim Hortons for Android (v7.1.487) – Static & Dynamic Analysis Reports (MobSF, AndroBugs, Manual Extraction). Internal forensic artifacts generated November 2025.
- Common Weakness Enumeration (CWE) – CWE-327: Use of a Broken or Risky Cryptographic Algorithm. MITRE Corporation.
- Common Vulnerability Scoring System (CVSS) v3.1 Specification Document. FIRST.org, 2025.

9.0 WORK DATE/HOURS LOG

Date	Student Name	Number of Hours	Description of Work
2025-10-20	Bright Ekeator	30 Minutes	Met with instructor to discuss project scope, tool selection, and proposal feedback.
2025-10-20	Bright Ekeator	1 hour	Team meeting to discuss project objectives, divide tasks, and plan analysis workflow for selected apps.
2025-10-24	Bright Ekeator	2 hours	Team meeting to set up the project proposal structure and created the GitHub repository for collaborative work.
2025-10-24	Bright Ekeator	3 hrs	Installed Andriller and set up test environment for Imo

			and Tinder APK extraction.
2025-10-27	Bright Ekeator	1 hour	Installed and configured Android Studio and AVD emulator with rooted image.
2025-10-28	Bright Ekeator	30 mins	Installed IMO app on emulator and confirmed package path
2025-10-29	Bright Ekeator	30 mins	Attempted adb backup on IMO; backup produced 1 KB file.
2025-10-30	Bright Ekeator	1 hour	Pulled internal IMO app data using adb pull.
2025-10-31	Bright Ekeator	30 mins	Organized project folders for analysis outputs
2025-11-01	Bright Ekeator	1 hour	Installed Python environment and Andriller; verified modules
2025-11-02	Bright Ekeator	1 hour	Extracted .ab backup using Andriller and decoded data
2025-11-03	Bright Ekeator	1 hour	Opened SQLite databases and reviewed message tables.
2025-11-04	Bright Ekeator	1 hour	Identified encrypted BLOB message content.
2025-11-05	Bright Ekeator	1 hour	Analyzed shared preferences for configuration data.
2025-11-06	Bright Ekeator	1 hour	Documented folder sizes and data extraction process.

2025-11-07	Bright Ekeator	1 hour	Drafted initial analysis report.
2025-11-07	Bright Ekeator	1 hour	Re-verified data integrity and repeated extraction steps
2025-11-08	Bright Ekeator	3 hours	Researched runtime key extraction and alternative decryption methods
2025-11-08	Bright Ekeator	2 hours	Reviewed all databases and performed additional test extractions.
2025-11-12	Bright Ekeator	30 mins	Tried Installing another APk file for Tinder and tried sign in but unsuccessful
2025-11-14	Bright Ekeator	30 mins	Prepared Progress report and compared previous analysis
2025-11-15	Bright Ekeator	2 hours	Attempted multiple Tinder APK installations (v14–v18), tested compatibility on emulator, collected crash logs.
2025-11-16	Bright Ekeator	1.5 hours	Research on Tinder root/jailbreak detection; attempted patching via apktool. **(Debugging time: 1 hr included)**
2025-11-17	Bright Ekeator	2 hours	Set up new emulator images (Android 11 & 12) to test APK compatibility.

2025-11-18	Bright Ekeator	1.5 hours	Used DCode to decode additional IMO timestamps; manually correlated values with call logs
2025-11-19	Bright Ekeator	2 hours	Deep dive into IMO shared preference XML files; recorded authentication token behaviour and storage paths.
2025-11-20	Bright Ekeator	3 hours	Full extraction test using Andriller TAR method; compared AB backup vs TAR output. **(Debugging time: 1 hr included — Andriller extraction errors)**
2025-11-21	Bright Ekeator	2 hours	SQLite DB Browser manual inspection of IMO call logs and message metadata; exported tables to CSV for report.
2025-11-22	Bright Ekeator	2 hours	Mapped Tinder cache directory contents; identified profile thumbnails and unmatched user artefacts.
2025-11-23	Bright Ekeator	3 hours	Attempted ADB-over-TCP connection between Windows host and Kali VM; resolved network bridging issues. **(Debugging time: 2 hrs)**
2025-11-24	Bright Ekeator	2 hours	Re-extracted IMO databases after

			emulator reset; validated consistency of timestamps across runs.
2025-11-25	Bright Ekeator	3 hours	Drafted Detailed Analysis section with screenshots; wrote correlation for call timestamps (UI → DB → DCode).
2025-11-26	Bright Ekeator	3 hours	Tinder forensic gap assessment: tested API-based message absence, confirmed no storage of unmatched swipes.
2025-11-27	Bright Ekeator	2 hours	Re-tested Andriller on Tinder TAR extraction; confirmed partial metadata recovery; documented database schema.
2025-11-28	Bright Ekeator	3 hours	Built Results & Insights section, focusing on manual DB Browser analysis and timestamp decoding. **(Debugging time: 1 hr — inconsistent timestamp formats)**
2025-11-29	Bright Ekeator	3.5 hours	Final report assembly
2025-11-29	Bright Ekeator	1 hour	Work Integration
2025-11-29	Bright Ekeator	1 hour	Video recording for submission
2025-11-29	Bright Ekeator	30 mins	Submission and push to GitHub.

2025-10-20	**Ifeoluwa Aribo**	**30 Minutes**	**Met with instructor to discuss project scope, tool selection, and proposal feedback.**
2025-10-20	Ifeoluwa Aribo	1 hour	Team meeting to discuss project objectives, divide tasks, and plan analysis workflow for selected apps.
2025-10-24	Ifeoluwa Aribo	2 hrs	Team meeting to set up the project proposal structure and created the GitHub repository for collaborative work.
2025-10-24	Ifeoluwa Aribo	2 hrs	Installed MobSF and AndroBugs; configured analysis environment for Tim Hortons app.

Date	Student Name	Number of Hours	Description of Work Done
2025-11-07	Ifeoluwa Aribo	1 hr	Installed and configured Android Studio and AVD emulator. Verified device connectivity using ADB commands.
2025-11-07	Ifeoluwa Aribo	0.5 hr	Identified installed packages on the emulator, including Reddit and Tim Hortons. Retrieved their package names and installation paths using ADB.
2025-11-07	Ifeoluwa Aribo	0.5 hr	Attempted to pull APKs from the emulator. Resolved path and directory permission issues.
2025-11-07	Ifeoluwa Aribo	0.5 hr	Created organized project folder structure on Kali for APKs, analysis reports, and tools.
2025-11-07	Ifeoluwa Aribo	1 hr	Set up Python environment for AndroBugs on Kali. Installed dependencies and validated Python 2.7 compatibility.
2025-11-07	Ifeoluwa Aribo	0.5 hr	Cloned AndroBugs_Framework repository from GitHub. Reviewed directory contents and verified script execution paths.

2025-11-07	Ifeoluwa Aribo	0.5 hr	Attempted to run androbugs.py with Python 3 and identified syntax errors. Verified successful execution using Python 2.
2025-11-07	Ifeoluwa Aribo	0.25 hr	Confirmed Python 2 installation and executed the AndroBugs help menu to verify arguments and analysis options.
2025-11-07	Ifeoluwa Aribo	1 hr	Conducted AndroBugs security scan on Reddit.apk. Collected analysis report and reviewed results.
2025-11-07	Ifeoluwa Aribo	0.5 hr	Analyzed Reddit scan results. Identified key issues including implicit intents, non-SSL URLs, and exported components.
2025-11-07	Ifeoluwa Aribo	0.5 hr	Summarized Reddit findings into a prioritized security risk report with recommendations.
2025-11-07	Ifeoluwa Aribo	1 hr	Conducted AndroBugs scan on Tim Hortons.apk. Generated and reviewed detailed analysis report.
2025-11-07	Ifeoluwa Aribo	0.5 hr	Compared security findings between Reddit and Tim Hortons APKs. Highlighted similarities and differences in vulnerabilities.
2025-11-07	Ifeoluwa Aribo	0.5 hr	Documented results, organized reports, and prepared summary for both APK analyses.
2025-11-07	Ifeoluwa Aribo	0.5 hr	Researched solutions for Python 3-compatible AndroBugs forks to improve future workflow.

Date	Name	Hours	Description
2025-11-21	Aribo Ifeoluwa	1.5 hrs	Installed MobSF and completed initial environment configuration.
2025-11-21	Aribo Ifeoluwa	0.5 hrs	Installed wkhtmltopdf and added the binary path to system and user variables.

2025-11-21	Aribo Ifeoluwa	1.5 hr	Investigated PDF generation error, located PDF module using project-wide search, identified missing pdfkit configuration.
2025-11-21	Aribo Ifeoluwa	0.5 hr	Added wkhtmltopdf configuration inside MobSF PDF handler and patched pdfkit.from_string call.
2025-11-21	Aribo Ifeoluwa	20 minutes	Restarted and validated MobSF after applying fix, confirmed PDF generation works.
2025-11-21	Aribo Ifeoluwa	1 hr	Uploaded Tim Hortons APK into MobSF and performed static analysis, reviewed permissions, code risks, and trackers.
2025-11-21	Aribo Ifeoluwa	1 hr	Uploaded Reddit APK and completed static analysis, reviewed findings and generated final PDF reports for both apps.

Date	Name	Hours	Description
2025-11-23	Aribo Ifeoluwa	4.0 hrs	**Team Collaboration - Andriller Debugging** - Spent 4 hours assisting Bright with troubleshooting Andriller tool issues, providing technical support and debugging expertise for mobile forensic analysis challenges
2025-11-25	Aribo Ifeoluwa	2 hrs	**APK Environment Setup** - Navigated to APKs directory, installed Reddit.apk, configured Android SDK manager for API 29, and installed multiple APK splits for both Reddit and Tim Hortons applications
2025-11-25	Aribo Ifeoluwa	2 hrs	**APK Extraction & Preparation** - Created dedicated directories for both apps, pulled base APKs and split configurations (en, mdpi, x86_64) from installed applications using ADB commands
2025-11-25	Aribo Ifeoluwa	3.5 hrs	**MobSF Dynamic Analysis Critical Issues** - Experienced severe connectivity problems: emulator repeatedly failing to connect to MobSF, ADB forwarding timeouts, port conflicts (8000, 1337, 7331), emulator disconnections, and persistent "Device not connected" errors despite multiple configuration attempts

2025-11-25	Aribo Ifeoluwa	1.5 hrs	**Dependency & Server Configuration** - Installed Python requirements, ran setup scripts, configured MobSF server on port 8001, and attempted various troubleshooting approaches for emulator connectivity
2025-11-25	Aribo Ifeoluwa	2.5 hrs	**Frida Alternative Approach** - After MobSF failures, pivoted to Frida framework: installed frida-tools, created custom PowerShell dynamic report collection script, and implemented SSL bypass with JavaScript hooks for traffic interception
2025-11-25	Aribo Ifeoluwa	2 hrs	**Network Traffic Interception Setup** - Configured mitmproxy for traffic capture, experimented with multiple proxy settings (127.0.0.1:8080, 10.0.2.2:8080), and attempted to bypass SSL pinning in target applications
2025-11-25	Aribo Ifeoluwa	1 hr	**Process Monitoring & Instrumentation** - Monitored running processes using ADB, identified package names, attempted Frida instrumentation on running applications, and troubleshooted various execution failures
2025-11-26	Aribo Ifeoluwa	1 hr	**MobSF Connection Breakthrough** - After extensive troubleshooting, successfully established stable connection between emulator and MobSF, resolving the persistent dynamic analysis connectivity issues
2025-11-26	Aribo Ifeoluwa	1.5 hr	**Dynamic Analysis Execution** - Performed comprehensive dynamic analysis on Reddit app through MobSF, monitored runtime behavior, API calls, and security vulnerabilities during execution
2025-11-26	Aribo Ifeoluwa	1hr	**Dynamic Analysis Execution -** Performed comprehensive dynamic analysis on Timhortons app through MobSF, monitored runtime behavior, API calls, and security vulnerabilities during execution
2025-11-26	Aribo Ifeoluwa	1.5 hr	**Report Generation & Documentation** - Generated detailed MobSF dynamic analysis report for Reddit app and Timhortons app, personally analyzed security

			findings, and compiled comprehensive DOCX analysis document with vulnerabilities and recommendations
2025-11-28	Aribo Ifeoluwa	1hr	Installed JADX GUI and JADX CLI. Verified Java JDK compatibility, configured PATH variables, and validated jadx-gui on test APKs. Tuned CLI flags for output readability, including deobfuscation options, class renaming, and line number retention.
2025-11-28	Aribo Ifeoluwa	1hr	Loaded the target APK into JADX for full decompilation. Generated Java source trees, resource trees, and smali references. Documented the internal package structure and located high-risk modules for manual review.
2025-11-28	Aribo Ifeoluwa	3hr	Performed deep Java code inspection using JADX outputs. Traced cryptographic API calls including Cipher.getInstance, MessageDigest, SecretKeySpec, and TrustManager overrides. Identified insecure primitives such as MD5 and SHA-1 and located the code paths that initialize insecure CBC mode encryption.
2025-11-27	Aribo Ifeoluwa	1hr	Analyzed exported Activities, Services, and Receivers inside the AndroidManifest.xml as rendered by JADX. Mapped exported components to their corresponding Java handlers and verified absence of signature-level permission protection.
2025-11-27	Aribo Ifeoluwa	1hr	Extracted the Tim Hortons Android app contents, located shared preference XML files and SQLite database files inside /data/data/digital.rbi.timhortons/. Performed full manual review of database schemas, table structures, and encryption indicators. Confirmed no sensitive data stored in SQLite databases. Validated SQL schema integrity, checked file permissions, and validated data segregation across multiple SQLite files.
2025-11-27	Aribo Ifeoluwa	1hr	Identified insecure storage of authentication tokens inside CapacitorStorage.xml. Extracted AWS Cognito access tokens, refresh tokens, and ID tokens. Decoded JWT structures, validated scopes and privilege levels,

			confirmed high-risk exposure. Mapped token flows against authentication model to assess privilege escalation risk.
2025-11-27	Aribo Ifeoluwa	1hr	Analyzed PII exposure across shared preference XML files. Identified email address, UUID, loyalty card data, and analytics identifiers. Performed correlation analysis between preference keys and backend API identifiers. Documented all exposed user identifiers and correlated them with threat scenarios.
2025-11-27	Aribo Ifeoluwa	1hr	Evaluated analytics identifiers stored in plaintext, including Adobe Experience Cloud IDs and Firebase Instance IDs. Mapped identifiers to potential tracking vectors. Assessed fingerprinting risk and cross-service correlation possibilities.
2025-11-27	Aribo Ifeoluwa	1hr	Assessed loyalty card data leakage, extracted barcode fields, cryptographic strings, card types, and identifiers. Mapped internal card properties to internal API operations.
2025-11-27	Aribo Ifeoluwa	1hr	Evaluated analytics identifiers stored in plaintext, including Adobe Experience Cloud IDs and Firebase Instance IDs. Mapped identifiers to potential tracking vectors. Assessed fingerprinting risk and cross-service correlation possibilities.
2025-11-27	Aribo Ifeoluwa	1hr	Compiled vulnerability rating for each issue using industry scoring logic. Classified Shared Preferences as HIGH risk, Authentication flow as HIGH risk, Encryption status as MEDIUM risk, and Database security as LOW risk.
2025-11-27	Aribo Ifeoluwa	2hrs	Prepared full technical remediation documentation. Recommended migrating to EncryptedSharedPreferences, enforcing Android Keystore storage, binding tokens to device traits,

			implementing short-lived tokens, applying certificate pinning, and realigning session management logic.
2025-11-27	Aribo Ifeoluwa	2 hrs	Created final consolidated report summarizing findings, impacts, risks, and remediation steps. Organized data into structured sections including Security Posture Summary, Technical Details, and Compliance Recommendations.
2025-11-29	Aribo Ifeoluwa	5.5 hours	Final report Compilation
2025-11-29	Bright Ekeator	1 hour	Work Integration
2025-11-29	Bright Ekeator	1 hour	Video recording for submission
2025-11-29	Bright Ekeator	10 mins	Submission and push to GitHub.