Exploring Data Extraction from Android Devices

What Data You Can Access and How

By Mattia Epifani - September 21, 2025

Source: ZENA FORENSICS - blog.digital-forensics.it

When I first started working in Digital Forensics, one of the fundamental **principles** was to always create a **bit-by-bit copy of the storage device**. Back then, we typically dealt with hard drives (internal and external), memory cards, and optical media.

The choices to be made—then as now—revolved around how to access the device, giving rise to the concept of **order of volatility**. Simply put, this means collecting the data that is most likely to disappear first. Traditionally, the main decision was whether to perform **live forensics** or to 'pull the plug' and create a disk image.

Today, both the world and digital forensics have changed significantly. Several factors now heavily influence how we approach data extraction:

1. **Solid-state storage**, often soldered to the device
2. Widespread deployment of **encryption mechanisms**
3. **Software-level access restrictions** designed to prevent full data reads

Key Guidelines and Terminology

Before tackling any case, analysts must consider several factors. Mistakes during acquisition can cascade through all subsequent steps. Several guidelines provide valuable insights:

- **SWGDE Best Practices for Mobile Device Evidence Collection & Preservation, Handling and Acquisition**
- **Requirements and Guidelines for a Complete End-to-End Mobile Forensic Investigation Chain** (CEN, 2022, ForMobile project)
- **Interpol Guidelines for Digital Forensics First Responders**

These documents introduce terminology now commonly used in mobile forensics, such as **AFU (After First Unlock), BFU (Before First Unlock), Hot and Cold**, and **Full File System (FFS) vs Logical acquisition**. Understanding these terms is crucial to follow proper workflows and mitigate risks associated with incorrect acquisition order.

This post is not intended to propose alternatives to these public-domain guidelines but to **compare the types of data obtainable under different acquisition methods**. No software comparison is made; rather, the focus is on **data accessibility regardless of specific tool capabilities**.

Device State and Unlock Code

The two main factors determining what can be extracted from a smartphone are:

1. The device's **state** (AFU or BFU)
2. The availability/knowledge of the **unlock code/pattern/password**

Based on these, we can define four scenarios:

1. Device is **BFU**, and unlock code/pattern/password is **unknown**
2. Device is **AFU**, and unlock code/pattern/password is **unknown**
3. Device is **AFU**, and unlock code/pattern/password is **known**
4. Device is **BFU**, and unlock code/pattern/password is **known**

For this post, we consider **standard Android devices**, excluding specialized cases such as:

- Devices with anti-forensics tools installed
- Devices managed by MAM/MDM systems with additional restrictions
- Devices running hardened OS variants (e.g., GrapheneOS)
- Devices using extra protection features like Android's 'Private Space'

Android Data Encryption (simplified)

Android devices implement **File-Based Encryption (FBE)** starting from Android 9, and it is mandatory from Android 10 onward. FBE encrypts files individually, enabling some to be accessed before user authentication through **Direct Boot**.

Files are categorized as:

- **Device Encrypted (DE):** accessible as soon as the device powers on (before user unlock)
- **Credential Encrypted (CE):** accessible only after user authentication

Additionally, **metadata encryption** protects filesystem metadata, ensuring that filenames and directory structures are secured along with the file contents.

For device states, the terms **AFU (After First Unlock)** and **BFU (Before First Unlock)** are critical:

- **BFU:** Only Device Encrypted files are accessible; Credential Encrypted files remain locked
- **AFU:** Both Device Encrypted and Credential Encrypted files can be accessed, assuming the device is unlocked or the unlock key is known

Data Extraction Scenarios

1. BFU Device, Unknown Unlock Code

In this scenario, tools capable of extracting **Device Encrypted files** are required. Some tools can also attempt **cracking the unlock code**. Cracking may occur on-device or offline after dumping memory, potentially moving the case to scenario 4 (**AFU + code known**) and enabling a **Full File System (FFS) acquisition**.

Free and open-source tools for BFU acquisition or cracking are extremely limited, especially for modern devices. Without appropriate tools, no data can typically be extracted (apart from basic device and OS information).

2. AFU Device, Unknown Unlock Code

Scenario 2 requires tools capable of **AFU acquisition**, accessing Credential Encrypted files without cracking the unlock code. Since Android encryption is binary (DE vs. CE), an AFU acquisition effectively yields a **Full File System (FFS)**.

3 & 4. AFU or BFU Device, Known Unlock Code

When the unlock code is known, data accessibility depends on the availability of tools that can **bypass Android restrictions**, often temporarily achieving **root access**.

- With such tools, a **Full File System (FFS)** extraction is possible
- Without such tools, a **Logical acquisition** is performed, using native communication protocols to extract data

Logical Acquisition Techniques

Logical acquisition interacts with the device using native protocols like **ADB** (requires Developer Options → USB Debugging) and **MTP**. There are at least six primary methods to extract data legally and effectively:

1. **Executing commands within the ADB shell**, including:

- Standard **Linux commands**
- **Logcat** – system and application logs
- **Getprop** – device properties
- **Package Manager (PM)** – app listing and info
- **Bugreport** – comprehensive system state
- **Content Provider** – access to structured app data

2. **Extracting content exposed via MTP**, typically the **Emulated SD Card**

3. **Creating an ADB Backup**, noting that on modern devices this usually **excludes native and third-party app data**, depending on the android:allowBackup parameter in the AndroidManifest

4. **Installing an agent**, which is essentially an APK that requests permissions to access native application data, particularly:

• Contacts

• Call Logs

• SMS/MMS

5. **Capturing screenshots via ADB**, enabling partial data collection from third-party apps. These screenshots can then be processed with image analysis to extract readable/searchable content

6. **APK Downgrade**, where a third-party app is temporarily replaced with a version that allows backup (android:allowBackup=true), enabling the creation of an ADB Backup for that app, before reinstalling the original version

These methods are particularly relevant even if a Full File System extraction is available, as they allow capture of volatile or easily lost data.

As always, the **order of data acquisition** should be determined by the **order of volatility** and the relevance of each potential data point to the investigation.

A relevant example is the publication *Android Dumpsys Analysis to Indicate Driver Distraction*, which demonstrates that data extracted using the commands **dumpsys wifi** and **dumpsys telecom** does **not survive a device reboot**.

Tools for Logical Extraction

Some known free and open-source tools include:

• **Android Triage** (github.com/RealityNet/android_triage)

• **AChoirX Android** (github.com/OMENScan/AChoirX)

• **Avilla Forensics** (github.com/AvillaDaniel/AvillaForensics)

• **Android Volatile Data** (github.com/mantal1/Android-volatile-data)

• **Android Quick Forensics** (github.com/botherder/androidqf)

• **Andriller** (github.com/den4uk/andriller)

• **Magnet Acquire** (magnetforensics.com)

• **dfAPKdngrader** (djangofaiola.blogspot.com/p/downloads.html)

Logical acquisition interacts with device protocols to extract **data accessible on an unlocked AFU device**. This includes information that would be lost if the device were powered off. Each tool may leverage one or more of the techniques listed above, executing commands, installing agents, or performing APK downgrades as needed.

Full File System Acquisition Techniques

Obtaining a **Full File System (FFS) acquisition** is significantly more challenging today without commercial tools. The major vendors and their products are well known, and, as expected, there is no single tool that works across all devices and scenarios. The choice of tools depends on operational needs, available budget, the most commonly analyzed scenarios (e.g., Android vs. iOS), and other factors.

The option to perform **independent rooting** without risking data loss (i.e., without triggering device wiping mechanisms) is now largely residual: it applies mainly to older devices or operating systems, previously modified or user-rooted devices, or specific chipsets with public vulnerabilities and exploits (e.g., **mtkclient**).

For Android devices, it is also essential to consider whether the available tool can **extract data stored in the KeyStore** (Android Keystore). Access to this information is crucial for applications that rely on it. KeyStore extraction is an **additional requirement** beyond obtaining a standard Full File System acquisition; some exploits may provide root access and allow extraction of the entire file system but **cannot access KeyStore data**.

Another topic gaining increasing attention is **RAM acquisition** from smartphones. For Samsung devices, the **Samsung Upload Client** (GitHub: bkerler/sboot_dump), an open-source tool developed by Bjoern Kerler, is available. Additionally, some commercial forensic tools have incorporated **RAM extraction capabilities** as part of their feature set.

Expected Acquisition Types

Summarizing the scenarios described above, during analysis it is reasonable to assume that the analyst may encounter one of the following acquisition types:

1. **BFU Acquisition** – obtained from a locked device
2. **Logical Acquisition** – obtained from an unlocked device using the various techniques mentioned above
3. **Full File System (FFS) Acquisition**
4. **Full File System (FFS) Acquisition with KeyStore data extraction**

Data Availability Across Acquisition Types

1. BFU Acquisition (Before First Unlock)

A **BFU acquisition** is obtained from a device that is **locked** and powered on or off. In this state, only **Device Encrypted (DE) files** are accessible. **Credential Encrypted (CE) files**, including app sandboxes and user media, remain inaccessible.

Despite these limitations, a BFU acquisition can provide valuable information for **device profiling** and initial investigation. Typical data available includes:

- **Basic device information:** brand, model, initialization date, IMEI, timezone, serial number, MAC addresses, chipset, OS version, security patch level
- **SIM cards and carriers:** ICCID and IMSI from files like telephony.db
- **Local accounts:** names and last login dates (e.g., 0.xml)
- **Accounts in use:** Google accounts, phone numbers, usernames (accounts_de.db)
- **Wi-Fi networks and PSKs:** stored in WifiConfigStore.xml
- **Installed applications and permissions:** from files such as packages.list, packages.xml, roles.xml, runtime-permissions.xml
- **Network statistics:** useful for profiling app usage (netstats folder)
- **Connected Bluetooth devices:** from bt_config.conf

Analysis of BFU data allows:

- Building a **timeline** from device initialization to the last activity
- Evaluating installed apps and their usage over time
- Assessing network connections via Wi-Fi and mobile traffic

A BFU acquisition can also inform decisions on whether **cracking the unlock code** is worthwhile. For instance, if the device has only limited relevance for the investigation, the time and resources required for code cracking might not be justified.

However, BFU acquisitions **cannot access user's personal data** stored in /data/ (app sandboxes) or /media/ (emulated SD card), as these are protected with **Credential Encryption**.

2. Logical Acquisition

A **Logical acquisition** provides **partial access to user data**, typically on an **unlocked AFU device**, using protocols such as **ADB** and **MTP**. The exact data obtained depends on the tool and techniques used. Key sources of information include:

- **Device info:** brand, model, initialization date, IMEI, timezone, serial number, MAC addresses, chipset, OS version, security patch level (via ADB commands)
- **Emulated SD Card:** extracted through MTP or ADB pull (usually /sdcard)

- **Installed applications and permissions**
- **Configured accounts**
- **Application usage information:** via dumpsys usagestats
- **Wi-Fi network information:** via dumpsys wifi
- **Connected Bluetooth devices:** via dumpsys bluetooth
- **Battery usage stats:** via dumpsys batterystats
- **Contacts, call logs, SMS/MMS, calendar:** through agent installation

Additional techniques:

- **Screenshots:** can capture app content visible on-screen, including some third-party apps
- **APK downgrade:** temporarily installs a previous version of an app that allows backup to extract data, then reinstalls the original version

Logical acquisition is **partial**: while it allows access to some user data and system info, it cannot extract app configuration files or databases in full, nor can it provide precise geolocation history beyond what is available via exposed system logs or backup.

3. Full File System (FFS) Acquisition

FFS acquisition is the most comprehensive method for devices using **File-Based Encryption (FBE)**, excluding volatile RAM data. It provides access to:

- **All Device Encrypted and Credential Encrypted files**, assuming AFU or known unlock code
- **Installed apps, native system apps, and their internal data**
- **System databases and configuration files**
- **Logs, network stats, and detailed usage metrics**

Effectiveness depends on the analyst's ability to process and interpret data, typically using multiple forensic tools. However, **FFS alone may not provide KeyStore data**, which is required for decrypting sensitive app data.

4. Full File System + KeyStore Extraction

When KeyStore data is accessible, the acquisition allows full decryption of files used by **secure applications**, such as:

- **Signal** (org.thoughtcrime.securesms)
- **Threema** (ch.threema.app)
- **Wickr Pro** (com.wickr.pro)
- **Samsung Health** (com.sec.android.app.shealth)
- **Samsung Rubin** (com.samsung.android.rubin.app)

This represents the **most complete dataset** an analyst can obtain, combining all FFS files with application-specific secured content.

Conclusions

The main goal of this article is to encourage analysts to consider all options and follow a **logical workflow** focused on **preserving and acquiring data relevant to the investigation**. There are no perfect guidelines, but existing standards help inform the best choices.

Key points to remember:

1. **Think before powering off a device.** Even with full access, shutting it down can cause irretrievable data loss.
2. **BFU acquisitions are essential for locked devices.** They help profile the device and provide insights for potential code cracking.
3. **Logical extraction may still be valuable**, even when FFS extraction is possible. Data obtained via tools like ADB can be easier to interpret, e.g., Battery Stats may be simpler to analyze than raw FFS data.
4. **Adapt workflows case by case**, considering available tools and expertise.

Appendix

Two tables summarize **data availability across acquisition types** (BFU, Logical, FFS, FFS + KeyStore):

Table 1: Operating System and Native App Files

This table shows which system and native application data is accessible in each acquisition type. Key findings:

- **BFU:** Limited to Device Encrypted files (device info, accounts, Wi-Fi, apps list)
- **Logical:** Partial user data via ADB/MTP (contacts, SMS, call logs, media)
- **FFS:** Complete system and app data
- **FFS + KeyStore:** Adds decryption capabilities for secure apps

Table 2: Third-Party Applications

This table illustrates third-party app data accessibility. Note that 'Logical' in the source material doesn't include 'screenshots' and 'APK downgrade' options, which can significantly expand data recovery from specific applications.

Key Observations:

- Third-party apps with **android:allowBackup=false** are not accessible via logical acquisition (standard ADB backup)

• **APK downgrade** technique can bypass this limitation for specific apps

• Apps using Android KeyStore (like Signal, Threema, Wickr) require **FFS + KeyStore** extraction for full data access

• **Screenshots** can capture visible app content but don't provide access to databases or configuration files