International Journal of Leading Research Publication (IJLRP)

E-ISSN: 2582-8010 ● Website: www.ijlrp.com ● Email: editor@ijlrp.com

Identifying Mobile Devices on Android Platform

Jagadeesh Duggirala

Software Engineer, Japan
jag4364u@gmail.com

Abstract

The need for unique mobile device identification on the Android platform is paramount for many applications, including user authentication, device tracking, fraud prevention, analytics, and personalization. However, due to privacy concerns, security requirements, and platform restrictions, obtaining a reliable and persistent device identifier has become more complex over the years. This paper explores various methods available on Android devices to identify unique mobile devices, from traditional hardware-based identifiers like IMEI to modern software-based techniques such as Advertising IDs and Google Firebase Instance IDs. It examines each approach's strengths, limitations, and practical use cases, helping developers make informed choices while balancing privacy and functionality.

Keywords: Android Applications, Identification, Fraud Prevention, Tracking, Authentication

1. Introduction

Mobile device identification has become a core feature in the development of mobile applications, whether for personalizing user experiences, enabling secure authentication, or implementing analytical tools. On Android, identifying unique devices can be challenging because of evolving privacy standards, especially with the introduction of Android 10 (API level 29) and later, which imposed stricter limitations on access to certain device identifiers.

This paper provides a comprehensive overview of different methods available to identify unique Android devices. It covers both hardware-based and software-based methods, their associated trade-offs, and the evolution of privacy policies. By the end of the paper, developers will have a better understanding of how to approach device identification on Android in a way that balances user privacy with functionality.

2. Hardware-Based Device Identifiers

Hardware-based identifiers are those that are embedded in the device itself, usually by the manufacturer. These identifiers are often considered more persistent and reliable, but due to privacy and security concerns, their access is increasingly restricted by the Android operating system.

2.1 IMEI (International Mobile Equipment Identity)

The **IMEI** is a globally unique identifier for mobile devices that connect to cellular networks. It is stored in the device's hardware and can be retrieved programmatically using the TelephonyManager API. This

identifier is tied to the device itself and remains consistent over time, making it ideal for tracking and identifying devices.

Advantages:

- Highly unique across all devices.
- Persistent and globally recognizable.
- Useful for fraud detection, device tracking, and anti-theft applications.

Limitations:

- Requires the READ_PHONE_STATE permission, which is subject to user consent.
- Starting with Android 10, apps can no longer access the IMEI without explicit permission, and access to IMEI is also restricted on non-phone devices (e.g., tablets).
- Privacy concerns as it is a personal identifier tied to the device.

Use Cases:

- Device tracking and theft prevention.
- Telecommunications and carrier-based services.

2.2 Android ID

The **Android ID** is a unique identifier generated when the device is first booted and is stored in the device's settings. It is a 64-bit string that remains consistent across app installations unless the device undergoes a factory reset.

Advantages:

- Easy to retrieve using Settings.Secure.ANDROID_ID.
- Does not require special permissions, unlike IMEI.
- Uniquely identifies the device within the Android ecosystem.

Limitations:

- The Android ID is reset after a factory reset, making it unsuitable for long-term device tracking.
- It may not be unique on certain custom ROMs or emulator environments.
- Not recommended for long-term user tracking as it can be reset by the user.

Use Cases:

- Personalized user experiences within apps.
- Analytics and session tracking where persistence across installs is not required.

2.3 Serial Number

The **serial number** is another hardware identifier that can be used to uniquely identify a device. This number is stored on the device's firmware and is accessible via the Build.SERIAL API.

Advantages:

- Unique per device.
- Persistent across app installations and reboots.

Limitations:

- Starting with Android 9 (Pie), the Build.SERIAL field requires the READ_PHONE_STATE permission, which may not be granted by the user.
- Access to the serial number is deprecated in some devices, and Android 10 and later may return an empty string.

Use Cases:

- Device-specific configurations or licensing.
- Internal use for device management in enterprise environments.

3. Software-Based Device Identifiers

Software-based identifiers are not physically tied to the device's hardware but are generated at the software level. These identifiers can be useful in scenarios where hardware-based identifiers are restricted or less reliable. They can also be more flexible and privacy-conscious.

3.1 Advertising ID

The **Advertising ID** is a user-resettable identifier provided by Google Play services for use in advertising and analytics. It allows developers to track users across different sessions or apps without using persistent device identifiers like IMEI.

Advantages:

- Unique and user-resettable, offering a balance between persistence and privacy.
- Widely used in analytics and advertising contexts.
- Does not require device-specific permissions to access.

Limitations:

- Users can reset the Advertising ID at any time through device settings, which may affect tracking consistency.
- Not suitable for use cases requiring absolute persistence.
- Requires Google Play services, limiting its use on non-Google-certified devices.

Use Cases:

- Ad targeting and analytics.
- Cross-app tracking for advertising networks.

3.2 Google Play Services Instance ID

The **Instance ID** is a unique identifier generated by Google Firebase services for each installation of an app. It is tied to the app instance and not the device, making it useful for identifying different installations of the same app across different devices.

Advantages:

- Unique per app installation.
- Ideal for managing user-specific data in cloud-backed apps (e.g., Firebase Cloud Messaging).

Limitations:

- Tied to app installations; the ID will change if the app is uninstalled and reinstalled.
- Not suitable for identifying the device itself; rather, it identifies the app instance.

Use Cases:

- Firebase-based applications requiring unique identifiers for user management.
- Push notifications and cloud messaging in apps.

3.3 UUID (Universally Unique Identifier)

The **UUID** is a standard for generating a unique identifier. In Android, it can be generated programmatically using the java.util.UUID class. The UUID is typically used for identifying app-specific instances or sessions, but it can also be used as a device identifier if stored locally.

Advantages:

- Easy to generate and implement.
- Does not require special permissions.
- Can be stored and used across app sessions.

Limitations:

- Not inherently tied to the device, making it unreliable for long-term identification of the device.
- Can be reset if the app data is cleared or the app is uninstalled.

Use Cases:

- Identifying user sessions or app instances.
- Tracking user activity across app sessions when tied to an app account.

4. Hybrid Methods: Combining Identifiers

In many cases, combining multiple identifiers can provide a more reliable and persistent device identification method. For instance, a developer might combine the Android ID with the Advertising ID to track user sessions while respecting privacy concerns. Using hybrid methods allows developers to adapt to various use cases while avoiding the limitations of individual identifiers.

For example:

- Combine the **Android ID** for device identification with the **UUID** or **Instance ID** for session management.
- Use the **Advertising ID** alongside **Google Play services** to track cross-app user interactions.

Advantages:

- Increases reliability and consistency in identifying users or devices.
- Reduces reliance on a single method that might be limited or restricted.

Limitations:

- More complex implementation.
- Potential conflicts between identifiers and privacy regulations if not handled carefully.

5. Statistics on the Impact of Device Identification in Fraud Prevention

Device identification has proven to be an effective tool in reducing mobile fraud. Several studies and reports provide insight into the significance of device identification in curbing fraudulent activities. Below are some relevant statistics:

5.1 Mobile Payment Fraud and Device Identification

According to statistics

Visa reported a **50% decrease in payment fraud** among merchants who implemented device fingerprinting and other device identification techniques in their payment gateways.

PayPal saw a **70% reduction in account takeover fraud** after integrating device recognition technologies into their authentication systems.

5.2 Device Identification and Two-Factor Authentication (2FA)

According to **Symantec's 2019 Internet Security Threat Report**, Combining device identification with traditional authentication methods (like 2FA) led to a **67% improvement in fraud detection rates**.

6. Conclusion

Identifying unique mobile devices on the Android platform remains essential for many use cases, but the landscape has changed significantly due to privacy regulations and platform restrictions. While hardware-based identifiers such as IMEI and Android ID are still valuable, they come with limitations and privacy concerns. Software-based solutions like the Advertising ID, Firebase Instance ID, and UUID offer more privacy-conscious alternatives, but they may not provide the same level of persistence or reliability.

Ultimately, the choice of device identification method depends on the specific needs of the application, as well as the balance between functionality, privacy, and security. Developers must stay updated on Android's evolving privacy policies and use a combination of identifiers to create robust solutions that comply with user expectations and legal requirements.

References

1. Android Developer Documentation: Settings.Secure
2. **Visa**, "Annual Fraud Prevention Report," 2019.
3. **PayPal**, "Security and Fraud Prevention: An Overview," 2020.
4. **Symantec**, "Internet Security Threat Report," 2019.