Your Signal, Their Data: An Empirical Privacy Analysis of Wireless-scanning SDKs in Android

Aniketh Girish
IMDEA Networks Institute / Universidad Carlos III de Madrid

Joel Reardon
University of Calgary / AppCensus

Juan Tapiador
Universidad Carlos III de Madrid

Srdjan Matic
IMDEA Software Institute

Narseo Vallina-Rodriguez
IMDEA Networks Institute / AppCensus

Abstract

Mobile apps frequently use Bluetooth Low Energy (BLE) and WiFi scanning permissions to discover nearby devices like peripherals and connect to WiFi Access Points (APs). However, wireless interfaces also serve as a covert proxy for geolocation data, enabling continuous user tracking and profiling. This includes technologies like BLE beacons, which are BLE devices broadcasting unique identifiers to determine devices' indoor physical locations; such beacons are easily found in shopping centres. Despite the widespread use of wireless scanning APIs and their potential for privacy abuse, the interplay between commercial mobile SDKs with wireless sensing and beaconing technologies remains largely unexplored. In this work, we conduct the first systematic analysis of 52 wireless-scanning SDKs, revealing their data collection practices and privacy risks. We develop a comprehensive analysis pipeline that enables us to detect beacon scanning capabilities, inject wireless events to trigger app behaviors, and monitor runtime execution on instrumented devices. Our findings show that 86% of apps integrating these SDKs collect at least one sensitive data type, including device and user identifiers such as AAID, email, along with GPS coordinates, WiFi and Bluetooth scan results. We uncover widespread SDK-to-SDK data sharing and evidence of ID bridging, where persistent and resettable identifiers are shared and synchronized within SDKs embedded in applications to potentially construct detailed mobility profiles, compromising user anonymity and enabling long-term tracking. We provide evidence of key actors engaging in these practices and conclude by proposing mitigation strategies such as stronger SDK sandboxing, stricter enforcement of platform policies, and improved transparency mechanisms to limit unauthorized tracking.

Keywords

Privacy, BLE, WiFi, Location, Beacons, Android SDKs

1 Introduction

Imagine walking through a shopping mall and, as you pass by stores, your phone buzzes with personalized ads and notifications. This seamless integration of the digital and physical worlds is powered by technologies like GPS, Bluetooth, and WiFi, as their sensing capabilities facilitate detecting device proximity or location. However, the use of these technologies raise significant privacy concerns because geolocation data is *inherently sensitive*: either coarse- or fine-grained geolocation data can reveal individuals' daily habits, work-home pairs [27], social structures [58, 133], and visits to sensitive sites like places of worship [19]. Unfortunately, the demand for location data by marketing, banking, and insurance firms has fueled a complex supply chain of actors specializing in location data collection, aggregation, and resale [74]. Companies like Azira [12] and Venntel [70] have been reported selling location data, including visits to sensitive locations such as abortion clinics [131] and even private sites like Jeffrey Epstein's island [129].

Despite the permission mechanisms implemented in Android and iOS to protect access to precise GPS data [37, 61], app developers and third-party SDKs exploit alternative channels to bypass these controls and obtain (or infer) users' location, often in the background without user awareness. One such mechanism is based on scanning wireless devices and beacon signals, including classic Bluetooth, Bluetooth Low Energy (BLE) beacons, and WiFi access points.[1] Such wireless scans can reveal users' geolocation as Bluetooth or WiFi beacons are tied to specific physical locations like stores, hotels, workplaces, or transit hubs [24]. Linking such geolocation data with user or device IDs like the Android Advertising ID (AAID) and MAC addresses make reversing users' identity and movement patterns straightforward [42].

The threats posed by the collection of wireless beacons as geolocation data proxies are real and severe, as several high-profile incidents and legal actions show. In 2014, the Snowden revelations exposed how devices could be tracked via MAC addresses through WiFi hotspots [91]. More recently, the Wall Street Journal reported that the FBI and ICE acquired location data through SDKs embedded in mobile apps [70, 119]. The Federal Trade Commission (FTC) has taken action against companies like X-Mode (now Outlogic) [29], inMobi, and Kochava for harvesting and reselling extensive location data without user consent, leading to serious concerns about constitutional rights in the U.S. [18–21]. A recent report by the German data protection authority reveals that apps collecting location data, including military personnel movements, was sold in data marketplaces, exposing intelligence agency sites, military bases, secret facilities and their personnel's habits [109, 128]. In the academic sphere, researchers demonstrated that wireless signal can be exploited to infer geographical locations and social structures [24, 51, 57, 85, 108]. Reyes *et al.* [107] found

[1]In this paper, we collectively refer to SDKs with BLE and WiFi scanning capabilities as "beacon SDKs". Equally, for simplicity, we refer to BLE and WiFi broadcast messages as "beacons."

Proceedings on Privacy Enhancing Technologies 2025(3), 361–381
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2025-0103

evidence of children-directed apps using WiFi APs as a proxy to GPS locations, violating the COPPA Rule. Similarly, Girish *et al.* [53] identified apps and SDKs leveraging local network scans to infer user location without user consent in smart home networks.

Despite these efforts, SDKs exploiting wireless scans for covert location tracking remain poorly understood. Consequently, existing app auditing mechanisms and privacy controls (including data randomization) have been proven insufficient to detect and protect users against such invasive practices, respectively [53, 122]. To fill this gap, we empirically analyze how 52 mobile beacon SDKs leverage GPS, BLE, and WiFi as side channels to covertly track individuals' proximity and locations, covering 9,976 Android apps. To uncover their scanning capabilities, data-sharing, and privacy risks, we develop a hybrid analysis pipeline that combines (i) static analysis, including API usage detection and control-flow analysis to study cross-library interactions and (ii) dynamic analysis, using an instrumented device that injects wireless events and signals at the OS-level to trigger SDK behaviors and observe app runtime behaviors. To the best of our knowledge, this study is the first large-scale empirical analysis of BLE and WiFi scanning SDKs in Android apps to systematically uncover their characteristics, interrelationships, and privacy implications. The key contributions of this work are:

- We identify 52 SDKs with WiFi and BLE scanning features integrated into at least 9,976 apps with an estimated cumulative install count of 55B devices. We find that beacon SDKs often offer a dual purpose, functioning both as beacon enablers and as analytics or advertising libraries. Specifically, 43 SDKs support analytics, 40 SDKs provide location services, and 9 SDKs integrate advertising or user profiling features (§5.1). Among them, we detect 28 SDKs that play a key role in extensive cross-library interactions, potentially enabling colluding SDKs to silently share with each other data such as Bluetooth, WiFi scans, and geolocation for advertising and tracking purposes (§5.2).
- We observe that 86% of beacon-enabled apps extensively collect personally identifiable information (PII), along with GPS coordinates, WiFi, and Bluetooth scan results (§6.2). However, their data collection practices extend further, with 19% of beacon SDKs engaging in ID bridging by linking persistent IDs (e.g., Android ID and MAC addresses) with resettable ones (e.g., AAID) to build detailed user mobility profiles, potentially violating Google Play Store policies (§6.3). These risks are exacerbated by the use of persistent proprietary IDs (e.g., Adobe's Marketing Cloud ID), unvetted cross-library data sharing, and by exploiting shortcomings of Android's permission and sandboxing models. Moreover, 71% of apps requesting location permissions fail to provide a rationale through Android's *shouldShowRequestPermissionsRationale()* API, leaving users unaware of why access is requested (§6.4). We also observe SDKs abuse vulnerabilities in unpatched devices to bypass the Android permissions governing BLE and WiFi permissions.
- These widespread practices undermine existing privacy protections and controls. In §7, we propose mitigation strategies, including: SDK sandboxing to restrict cross-library data sharing, performing runtime audits to detect unauthorized tracking, enforcing stricter platform policies to curb ID bridging, and improving transparency mechanisms by requiring clear permission rationales and explicit disclosures of SDK data access practices. Additionally, we advocate for stronger regulatory oversight, platforms' permission re-designs and stricter vetting processes (i.e., Google Play Protect) to proactively conduct large-scale audits and enforcement actions against non-compliant tracking practices.

Responsible disclosure. We shared a preprint of this paper with Google as operator of the Android platform, and with the European Data Protection Supervisor (EDPS), the Spanish Data Protection Agency (AEPD), and the French National Commission on Informatics and Liberty (CNIL).

Research artifacts. To promote transparency, reproducibility, and future research, we will release the final dataset, beacon detection scripts, static analysis pipeline, and analysis code used in this study, excluding proprietary components. These artifacts will be publicly available at: https://github.com/wireless-scanning-SDKs/.

2 Background

This section provides an overview of BLE (§2.1) and WiFi (§2.2) beacon technologies as location proxies, and an overview of the existing Android permissions for restricting apps' access to BLE and WiFi scanning capabilities (§2.3).

2.1 BLE Beacons

A BLE beacon is a non-pairing device broadcasting unique IDs like UUIDs or MAC addresses to infer proximity or location. Unlike GPS-based positioning systems, BLE beacons are deployed in fixed locations—e.g., stores, metro stations or offices—so that they provide fine-grained indoor location for like retail [68], healthcare, logistics, or crowd management at events [89]. These fixed signals are collected alongside GPS coordinates and are often aggregated into publicly accessible databases such as Wigle [127], and proprietary ones. This allows SDKs to infer a user's location solely through nearby BLE signals and access to such datasets.

BLE beacons are not a Bluetooth SIG standard; rather, they are custom standards developed by large providers or groups of companies. Popular BLE beacon solutions include iBeacon by Apple [63], the now deprecated Eddystone by Google [64], and the widely supported open-source and platform-agnostic AltBeacon [4], which is becoming a predominant solution due to their cross-platform interoperability. We note that standard Bluetooth devices can be also scanned to collect device names and MAC addresses, which could be used for environmental sensing (*e.g.,* by inferring locations based on paired BLE devices), and social network inference [81].

2.2 WiFi Beacons

WiFi Access Points (APs) broadcast beacon frames containing their MAC addresses and received signal strength indicator (RSSI) over 802.11b/g/n channels. Mobile devices can passively scan these beacons to locate and connect to nearby APs by sending association frames. WiFi APs are typically deployed in known locations (*e.g.,* homes, shops, hotels, and airports) and rarely move once deployed. This means that correlating WiFi SSID and BSSID data[2] with free online databases (e.g., wigle.com) or commercial location services

[2]SSID is the name of a wireless network that devices use to identify and connect to it. BSSID is the unique identifier for a specific access point within a wireless network (i.e., MAC address).

(*e.g.,* here.com) makes them highly effective for determining user location. In fact, there are commercially-available WiFi Positioning System (WPS) that facilitate inferring users' locations from WiFi signals, as a complement or alternative to GPS-based systems. Apple integrates WPS [7] into iOS devices for enhanced location services, though Rye *et al.* [110] recently highlighted the potential for mass surveillance using this data. Similarly, Google maintains a WPS populated through WiFi AP data collected from Android devices and its ecosystem, which also serves as a data source for Android's coarse location permission [55, 121]. These WPSes expose APIs that allow a client to submit WiFi scan data and receive location estimates in return. Other providers, such as Skyhook Wireless [114] and Navizon [88], offer hybrid systems combining GPS, WiFi, and cellular signals to triangulate device location.

2.3 Permission Overview

Android's permission model regulates access to WiFi and Bluetooth scanning functions to protect user privacy and security [36]. Over successive releases, Android's permission model has evolved to impose stricter controls on BLE and WiFi scanning capabilities, particularly due to their dual usage as location proxies.

Bluetooth. Initially, Bluetooth operations required only a generic permission for scanning and connecting with nearby devices. However, starting with Android 6, apps also had to request the runtime[3] `ACCESS_COARSE_LOCATION` permission, as BLE scans could indirectly reveal location data [32]. This was later upgraded in Android 10, mandating the `ACCESS_FINE_LOCATION` permission for Bluetooth scanning due to its increased risk of exposing the precise location data [71]. However, this security improvement only led to confusion among users, especially when apps that did not inherently need location data—such as those for connecting to Bluetooth peripherals—were also forced to request location permissions [130]. To address this issue, Android 12 introduced a finer-grained model with three new permissions: one for scanning (`BLUETOOTH_SCAN`), one for establishing connections (`BLUETOOTH_CONNECT`), and one to advertise as a peripheral (`BLUETOOTH_ADVERTISE`). Developers must request both `ACCESS_FINE_LOCATION` and `ACCESS_COARSE_LOCATION` to allow users to choose between precise or approximate location access at runtime. Yet, without clear context, users may unintentionally grant excessive location privileges—e.g., precise location for Bluetooth pairing. To mitigate this, Android 12 introduced the `neverForLocation` flag in 2021 to ensure that Bluetooth scan data is not misused for location inference.

WiFi. Since Android's first release, apps using WiFi must request the `CHANGE_WIFI_STATE` and `ACCESS_WIFI_STATE` permissions to toggle connectivity, modify settings, and provide connection details. Since Android 8, any app that scans the WiFi network is also required to request at least one location permission (i.e., `ACCESS_FINE_LOCATION` or `ACCESS_COARSE_LOCATION`). More recently, Android 13 introduced a new runtime permission, `NEARBY_WIFI_DEVICES` for connecting to nearby devices via WiFi. Android 13 also added support for the `neverForLocation` tag to specify WiFi scan data is not used for location inference.

[3]Unlike normal permissions, which are granted during installation, *runtime* or *dangerous* permissions protect sensitive resources and data (*e.g.,* location) and require explicit user approval at runtime [36].

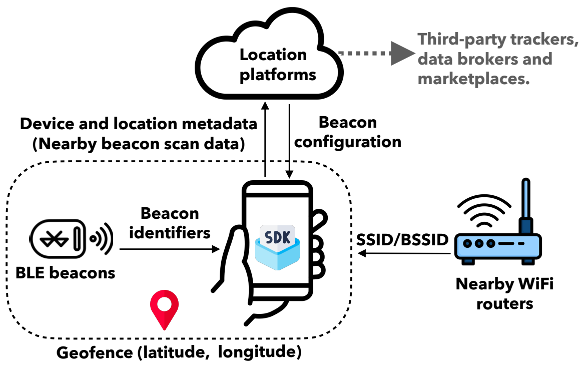
Location platforms
Third-party trackers, data brokers and marketplaces.
Device and location metadata (Nearby beacon scan data)
Beacon configuration
Beacon identifiers
SDK
BLE beacons
SSID/BSSID
Nearby WiFi routers
Geofence (latitude, longitude)

Figure 1: Beacon data ecosystem.

Platform Fragmentation. Due to the high version fragmentation in the Android ecosystem, many apps and SDKs intentionally exploit OS vulnerabilities and old (typically less restrictive) versions of the Android permission model to perform BLE and WiFi scans. Another key limitation of the permission model is its inability to separate between permissions requested by the host app and those by third-party SDKs [49], even for runtime permissions that grant access to sensitive data. As a result, SDKs can piggyback on the permissions requested by the host app for secondary purposes such as surveillance or advertising.

3 Threat Model

We consider an adversary whose primary objective is to collect geolocation data from mobile Android apps by scanning nearby devices using Bluetooth or WiFi permissions (see Figure 1). This adversary can be any party with programmatic access to the device for scanning data, including the app developer (the first party) as well as third party SDKs offering advertising, analytics, or location-based solutions. We consider three key threats posed by such adversaries:

(1) **Continuous Location Tracking through Proxy Sensors.** Wireless-enabled location SDKs or Android apps can *silently* track users by continuously scanning for nearby WiFi and BLE signals. Scan results can include: device information (e.g., device name), network information (e.g., SSID/BSSID, MAC addresses), and other unique device IDs as in the case of BLE beacons. To infer user location, wireless scanning data can be correlated with external databases that map MAC addresses, beacons and WiFi AP BSSIDs and SSIDs to geographic coordinates as described in the previous section. SDKs often enrich scan data with precise GPS coordinates and other device or user identifiers like the AAID to construct detailed mobility profiles of millions of users. [4] Table 7 in the Appendix provides an exhaustive description of the PII types and user/device IDs considered

[4]The Android OS allows app and SDK developers to programmatically access a wide range of IDs with varying properties [38]. The MAC address of a WiFi AP are device-specific, persistent and globally unique IDs. Similarly, the user's email address, even if hashed, is a user-specific, persistent and unique ID, unless it is shared by a group of users. The AAID is a device-specific, resettable and globally unique ID. The Firebase ID, instead, is a globally unique and app-specific ID that is resettable by reinstalling the app to which it is scoped.

in this paper, categorized by their nature. These rich user profiles may reveal sensitive information such as social structures, frequently visited locations and lifestyle patterns [42].

(2) **ID Bridging.** Beacon information facilitates collecting unique user geolocation fingerprints—as demonstrated by de Montjoye [27]—due to the uniqueness of human mobility patterns. Mobility fingerprints not only allow distinguishing individuals even in anonymized datasets but also enable ID bridging, i.e., correlating and bridging users data and fingerprints for continuous user tracking and profiling. These methods, particularly when used to bridge resettable IDs like the AAID, significantly undermine user attempts to maintain anonymity on the Internet, even if users opt-out of personalized ads through system settings or by resetting their AAID. To mitigate these privacy risks, Google has set various policies and best practices to inform and educate app developers about which IDs would be better for specific purposes [99]. Google strictly prohibits linking resettable IDs (*e.g.,* AAID) with persistent or global IDs for advertising or analytics purposes without sufficient transparency [38, 99]. §6.3 analyzes ID bridging practices across beacon SDKs, including AAID bridging, while §5.2 shows how colluding SDKs programmatically interact with each other to synchronize user profiles without user awareness. Oftentimes, geolocation data is aggregated and sold to advertisers, analytics companies, and other entities like data brokers, causing significant privacy harm to users.

(3) **Exploitation of Outdated Permission Models.** SDKs can exploit vulnerabilities in Android's permission model (§2.3) to access sensitive data in unpatched devices. Methods to achieve this include piggybacking on permissions granted to the host app, using legacy APIs with less stringent requirements, or leveraging side channels to bypass restrictions and infer information without explicit permission requests. This type of attack is facilitated by Android's version fragmentation: according to Statista, over 41% of Android devices run version 12 or below as of January 2025. §6.2 reports evidence of SDKs using these techniques to access location data.

4 Methodology

Figure 2 summarizes the methodology we developed to achieve the paper objectives. We combine both static and dynamic analysis techniques to comprehensively detect and analyze how apps collect geolocation data through BLE and WiFi beacons, its dissemination with user IDs, and the SDKs offering such solutions. We note that, due to the inherent limitations of black-box testing, we do not claim completeness. However, our methods provide actual evidence of privacy concerns associated with the use of wireless scanning capabilities by SDKs. We next describe each of the components of our methodology.

4.1 Dataset

We use the AndroZoo dataset [3] to collect 1,008,539 apps uploaded after January 2023. AndroZoo is a large-scale collection of Android apps that includes over 10 million items historically indexed on the Google Play Store. We filter this dataset to select 574K apps available in the Google Play Store when accessed from the European Union (EU) as of July 2024, ensuring that our study covers the most recent app versions. We use an open-source Google Play scraper [96] to collect metadata for each app, including its self-disclosed app category, developer information, data safety labels, and install counts.

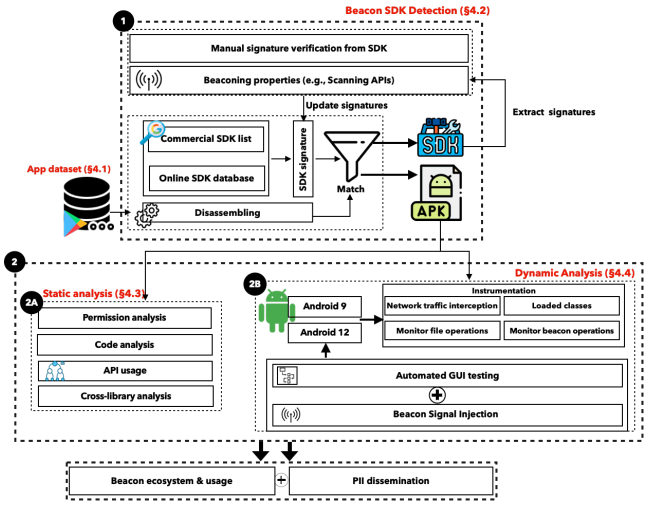

Figure 2: Methodology overview. Processes 2.a and 2.b. run in parallel.

4.2 Detecting Beacon SDKs

There is no publicly available inventory of SDKs offering BLE and WiFi scanning capabilities. While some efforts document advertising and analytics SDKs [49, 82, 103, 104, 134], beacon-enabled SDKs remain under-reported, often embedded within broader functionalities, making them difficult to detect and analyze. To address this gap, we create a first-of-its-kind public dataset, capturing a representative set of widely used SDKs with wireless scanning implementations. Our approach systematically identifies these SDKs through a multi-step iterative process, combining third-party datasets and manual searches to curate a structured list of SDKs actively engaging in wireless scanning.

- **Step 1: Identifying Known SDKs.** To identify SDKs that perform BLE and WiFi scanning, we begin with an initial set of SDKs listed by Exodus [103] and Libradar [82], two widely used SDK detection tools. While these tools cover popular advertising and analytics SDKs and label them based on their capabilities (including BLE and WiFi capabilities), they are not comprehensive, potentially missing newer or less-documented SDKs as well as scanning-specific SDKs. From these sources, we identify an initial list of 14 SDKs related to BLE/WiFi scanning.
- **Step 2: Targeted Online Searches.** We expand the initial list of SDKs identified in Step 1 by conducting web searches for commercial SDKs offering proximity, location, and Bluetooth/WiFi scanning functionalities. For that, we perform targeted keyword

searches using Google's search engine. [5] This process identified 22 additional SDKs that were not included in the initial list. However, some results are false positives triggered by keyword matches while the SDKs do not implement beacon scanning. Hence to confirm functionality, we manually review official documentation, API references, integration guides, and discussions in SDKs' developer forum, along with source code repositories (where available) to identify SDKs that explicitly invoke BLE or WiFi scanning capabilities. [6]

- **Step 3: Refining & Expanding SDK Selection Through Static Analysis.** We refine and expand the set of SDKs populated in the previous steps, by applying the static analysis techniques illustrated in §4.3 to automatically analyze a subset of 362K apps invoking methods indicative of beacon scanning. These features are extracted from the curated list of SDKs identified in Steps 1 and 2, producing *SDK signatures* for each SDK that cover from system API calls related to BLE and WiFi scanning (e.g., 'WifiManager.startScan()' [35, 39]) to SDK attribution signals such as class names, URL endpoints, and custom permissions. Additionally, we search for special functionalities such as UUID broadcasting and RSSI analysis, which further indicate BLE/WiFi scanning behavior. For reference, an example set of signals is provided in Table 9 in the Appendix. After applying our refinement process, we identified a total of 55 SDKs.

Validation. To ensure high-confidence detection and minimize false positives, we manually analyze the code of up to three representative apps per SDK to confirm whether these SDKs implement scanning capabilities. This check allowed us to remove three SDKs designed for general connectivity with peripherals rather than for wireless scanning. At the end of this manual process, we curate a final signature set of 52 beacon SDKs features and signals, which we use to statically identify at scale apps integrating these SDKs.

4.3 Static Analysis

We apply static analysis to examine how apps integrate and interact with identified beacon SDKs. Specifically, we look for unique SDK signatures (§4.2) to identify their presence across apps, including app manifest metadata (declared permissions, services, and intent filters) and code-level attributes. We also inspect execution paths via control flow analysis to determine how apps invoke, interact with, or delegate scanning capabilities to other integrated SDKs, as we describe next. Our static analysis follows accepted research practices, focusing on studying public code structures and API calls, ensuring that no proprietary insights or trade secrets are extracted [10, 52, 100, 124].

Permission Analysis. We use a custom-built parser to analyze the manifest of each app, extracting AOSP and custom permissions requested by apps and beacon SDKs, along with their declared services, providers, and receivers. We also analyze metadata such as the `neverForLocation` attribute which indicate whether location permissions are used exclusively for non-location purposes, and the `shouldShowRequestPermissionRationale()` API, which provides an educational UI explaining why a specific permission is needed to enable a feature. App developers should declare these attributes to ensure transparency so their absence can indicate a lack of proper disclosure. These elements are studied in §6.4 to assess beacon SDKs' transparency over data collection.

Code Analysis. We use Androguard [5] to extract Dalvik bytecode and identify SDK signatures (*e.g.,* class names and APIs) that indicate beacon SDK functionality and data dissemination. These signatures are detailed in Table 8. To analyze execution paths, we construct control flow graphs (CFGs) for each app, where nodes represent methods or functions (with associated class names), and edges denote function calls between them. CFGs trace execution from source methods (e.g., Android lifecycle methods like onCreate() and onResume(), where SDKs typically initialize) to sink APIs that collect sensitive data, such as location (e.g., LocationManager.getLastKnownLocation()) or beacon data (e.g., WiFi scan results). A mapping of API calls considered in this process is provided in Table 10. To attribute SDKs and identify SDKs actively being used rather than dead code, we match class names in CFG nodes against known SDK signatures extracted from decompiled app code.

API Usage. To analyze how beacon-enabled apps interact with sensitive resources, we traverse the CFG to trace data flows. Starting from sink APIs (e.g., 'LocationManager.-getLastKnownLocation()'), we visit nodes backward to locate the source methods invoking these APIs. This approach links sensitive API calls to the Android permissions protecting them, allowing us to study the data flows within the app and also third-party SDKs that may piggyback on these permissions (§6.1). As an official and updated comprehensive permission mapping database is not available, we construct one by combining: (1) pre-existing permission mappings from PScout [11], Androguard, and Axplorer [13]; and (2) mappings derived from the `@RequiresPermission` annotations in the AOSP source code, which we publicly release as part of our research artifacts.

Cross-Library Analysis. We statically investigate interactions between SDKs embedded within the same app to identify dependencies and data-sharing practices between co-located SDKs. Specifically, we identify cross-library interactions by studying function calls where one SDK code invokes methods or accesses data from another SDK. We attribute these interactions among SDKs by detecting function calls between classes associated with beacon SDKs. To distinguish cross-library interactions from app-internal calls, we compare the class names of the caller and callee classes: if their top- and second-level domains (1) do not match each other, and (2) do not match the host app's package name, the interaction is flagged as cross-library. We manually validate all detected interactions to avoid mis-reporting cross-SDK interactions.

4.4 Dynamic Analysis

Dynamic analysis allows us to gather actual evidence of beacon SDK runtime behavior by executing apps on instrumented devices [1, 78, 105, 107]. Specifically, we observe how apps interact with sensitive resources, handle beacon data (e.g., BLE, WiFi), transmit network traffic, and respond to injected BLE and WiFi signals. To achieve this, we use a device farm comprising eight Android 9 and eight Android 12 Google Pixel 3a smartphones located in the EU, each running an instrumented version of the operating system. This

[5] We use keywords such as "beacon SDK," "proximity SDK," "BLE SDK," and "WiFi beacon SDK."

[6] We manually identified more than 200 system APIs related to BLE and WiFi scanning from Android's official API reference and AOSP (Android Open Source Project) source code. These methods are released as part of our open-science approach.

dual-version approach allows us to analyze how app behaviors adapt to changes in the Android permission model across different versions and how SDKs may abuse legacy versions.

Instrumentation. We use a customized AOSP version to transparently monitor runtime resource access, such as accesses to permission-protected APIs, BLE/WiFi scans, I/O file operations, and capturing all network traffic, by instrumenting the relevant methods and APIs to capture their activity. We observe reads and writes to TLS sockets by instrumenting relevant APIs, allowing us to analyze network traffic without interfering with the TLS handshake or using our own certificates. We also decode common encodings like gzip and base64, along with bespoke obfuscation methods used by popular SDKs (e.g., Forter, JPush, and Yandex) to identify disseminated sensitive data, as discussed in §6.2. To complement our static analysis-based SDK detection (§4.3), we instrument the Android Runtime (ART) to log classes loaded at runtime by tracking the `FindClass` method of the class linker to detect classes loaded during execution. This approach effectively increases the reliability of static signals and enables the identification of obfuscated classes that static analysis might miss.

Automatization. We use Android Monkey [6] to automate app execution with synthetic UI inputs for 8-10 minutes. To bypass registration walls, each device is configured with unique pseudonymous identifiers (e.g., phone numbers, email addresses, and usernames), allowing us to trace the potential dissemination of this data in network flows for ID bridging. Our automation also handles standard logins, including Single Sign-On (SSO) flows like Google SSO whenever applicable.

Beacon Signal Injection. In addition of automatizing UI inputs, it is essential to simulate realistic beacon signals to trigger runtime responses in SDKs implementing them. We achieve this by injecting Bluetooth and WiFi scan results during app scans, simulating nearby devices or networks directly on our AOSP instrumentation. The injected BLE advertising data conforms to standard beacon formats and includes beacon types such as iBeacon, AltBeacon, Eddystone (URL, UID), and GAEN beacons, formatted according to their respective standards [4, 62–65]. Additionally, we use a Raspberry Pi 4 to replay intercepted BLE beacon payloads and MAC addresses every 5 seconds, with the goal of exercising further app activity upon beacons detection. Spurious WiFi networks are also injected, including SSIDs and BSSIDs, as well as SSIDs appended with `_nomap` and `_optout`, to evaluate whether SDKs respect router-owners preferences to exclude their devices from scanning. We use distinctive palindromic MAC addresses in the injected results to monitor their dissemination in TLS flows to the cloud.

4.5 Limitations

As with any empirical measurement study, our analysis is inherently constrained by the black-box nature of testing methods, the opacity of SDKs, and the continuous evolution of implementations. In fact, our list of beacon SDKs extracted with static analysis methods may not offer a complete picture of this complex landscape due to code obfuscation, reflection, and dynamic loading [22, 47, 84].

Dynamic analysis successfully executes 97% of the apps, despite challenges like root detection, certificate pinning, emulator checks, and behavioral fingerprinting, which may limit visibility into SDK behaviors. Beacon injection effectively triggers scanning in most cases but may not trigger SDKs relying on custom or proprietary formats that are not publicly documented. Additionally, Android Monkey automates standard interactions, though it cannot bypass CAPTCHA or 2FA logins yet prior work [107] shows it explores code paths 60% similar to human interaction.

We run our experiments on EU-based devices where stricter privacy rules apply so it is possible that specific SDK behaviors may vary in jurisdictions with more permissive regulations. Nevertheless, these limitations do not compromise the validity of our findings: while we do not claim completeness, our methodology effectively uncovers previously undocumented tracking behaviors within beacon SDKs, highlighting systemic privacy risks and the need to regulate their usage.

Table 1: Top 20 SDKs in the dataset, classified depending on their beacon types: Bluetooth, WiFi or they operate as integration partners. For the purpose of each SDK, in addition to analytics and location, we mark SDKs that offer advertising (●) and profiling (◇) services.

SDK Name	# Apps	Total Installs	Beacon Type			Purpose Type	
			BLE	WiFi	Integration	Analytics	Location
AltBeacon	4,022	5B	✓			✓	✓
Adobe Experience Platform	1,328	8B			✓	✓	
Kochava ●	1,117	15B		✓	✓	✓	✓
Salesforce Marketing Cloud ●	1,080	6B			✓	✓	✓
Estimote	510	201M	✓			✓	✓
LeanPlum ◇	456	8B	✓	✓		✓	✓
Gimbal ●	396	3308M	✓	✓		✓	✓
Radius Networks	369	359M	✓			✓	
mParticle	367	2B			✓	✓	
Ad4Screen ●	198	1B	✓	✓			
Kontakt	195	31M	✓			✓	
CueAudio	190	9M				✓	✓
Swrve ◇	153	2B	✓			✓	✓
Reveal Mobile	109	6M	✓	✓		✓	✓
Exponea	99	191M		✓	✓	✓	
Radar	93	482M	✓			✓	✓
IndoorAtlas	92	7M	✓	✓		✓	✓
SignalFrame	89	56M	✓	✓			✓
Bazaarvoice	88	420M			✓	✓	✓
Huq Sourcekit	81	347M		✓	✓	✓	✓

5 Beacon SDK Landscape

This section studies the beacon SDK landscape, focusing on their purpose, market share (§5.1), and integration capabilities to leverage complementary features and data sharing (§5.2).

5.1 SDK Purposes and Prevalence

Using our SDK detection method (§4.2), we find the 52 SDKs with beacon capabilities embedded in at least 9,976 apps collectively installed on 55B devices.[7] The market share and feature set of beacon SDKs varies significantly. Table 1 lists the top-20 SDKs categorized by beacon type (BLE, WiFi, or integration partners) and their primary and secondary purposes (*e.g.*, analytics, location services, or advertising). A full list is provided in Table 6 in the Appendix. The most widely used SDKs are AltBeacon (40% of apps), Adobe Experience Platform (13% of apps), Kochava (11% of apps), Salesforce Marketing Cloud (11% of apps) and Estimote (5% of

[7] Google Play install counts provide an approximate range of market reach, not exact adoption figures. Prior work has used them as a standard reference for estimating user base and impact, though they have limitations such as the exclusion of pre-installed or sideloaded apps [123].

Table 2: Top 10 most common SDK combinations. Count is the number of apps embedding each combination.

SDK Combination	Count
(AltBeacon, Kontakt)	270
(AltBeacon, Radius Networks)	228
(CueAudio, Gimbal)	173
(AltBeacon, Areametrics, Cuebiq, Reveal Mobile)	117
(Adobe Experience Platform, Gimbal)	100
(AltBeacon, Cuebiq)	80
(LeanPlum, mParticle)	78
(AltBeacon, Salesforce Marketing Cloud)	73
(AltBeacon, Estimote)	67
(AltBeacon, Cuebiq, SignalFrame)	59
Total Unique SDK Combinations	281
Total Installs	6B

apps),[8] which are collectively embedded in apps with a total of 37 billion installs. While we find beacon SDKs in apps from every Google Play store categories, they are more prevalent in Lifestyle (12%), Shopping (9%), and Sports (8%), with BLE and WiFi beacon SDKs being more prominent in specific market sectors:

- BLE beacon SDKs are found in 54% of the apps with a total install count of 55B devices. They dominate the Lifestyle (20%) and Business (12%) app categories, with SDKs like Radius Networks (44%) and Estimote (34%) more likely integrated in Lifestyle apps. Interestingly, all BLE beacon SDKs also support geofencing for location-based services, including targeted advertising and proximity marketing (*e.g.,* Radius Networks and Estimote [44, 90]).
- WiFi beacon SDKs are found in 17% of the apps, totaling 72B installs. They are commonly found in Sports (16%) and News & Magazines (10%) apps. The SDKs Yinzcam Sobek and Gimbal SDKs [67] specialize in sporting events and live stadium experiences. Anecdotally, some apps in these categories also embed audio beacon SDKs (*e.g.,* CueAudio) to synchronize location data with contextual audio signals, enhancing real-time interaction and audience tracking [23].

Capabilities. Some of the beacon SDKs identified like Kochava or Adobe Experience Platform are well-known advertising and tracking services [104]. We analyze the capabilities offered by beacon SDKs by inspecting their public documentation (see §4.2). We observe that most of them claim to offer multiple features to app developers, primarily analytics services (43 SDKs, with a total install count of 52B installs), location services (40 SDKs, 41B installs), advertising (9 SDKs, 21B installs), and user profiling (5 SDKs, 11B installs). These features are not mutually exclusive. For example, while the the main business area of Kochava, Salesforce Marketing Cloud, and Adobe Experience Platform is an advertising, analytics and identity graph services, respectively, they also integrate beacon scanning capabilities in some of their SDK versions to enhance their analytics and advertising businesses with location data. This increases the privacy risks for consumers.

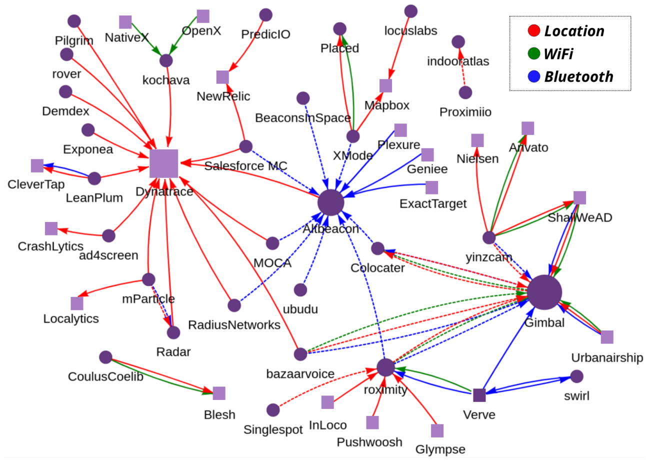

Figure 3: Cross-library interactions between beacon and non-beacon SDKs (square nodes). Dotted lines represent interactions between beacon SDKs, while solid lines show interactions with non-beacon SDKs. Arrows indicate directionality, from caller to callee SDK, with colors denoting the specific APIs accessed.

5.2 Cross-Library Analysis

The 9,976 analyzed apps also include 331 non-beacon SDKs with further data collection capabilities, including advertising and tracking services. We study the interactions between beacon SDKs with other SDKs (including non-beacon SDKs) that are co-located within the same app using the methodology outlined in §4.3. Such co-existence allows SDK operators to exchange sensitive data or complement their functionality including their tracking capabilities. In total, we identify 281 unique SDK combinations across the 9,976 apps containing at least one beacon SDK. Table 2 reports examples of such combinations, being AltBeacon and Kontakt the most common combination, found in 270 apps (195M users).

Figure 3 captures SDK cross-invocations and data flows. We detect 28 beacon SDKs exposing APIs for cross-library interaction and data sharing to others. We note that location APIs (red) are the most frequently invoked, often transmitting data from beacon SDKs to advertising and analytics services. WiFi-based APIs (green) are used for network-based geolocation, while BLE APIs (blue) enable proximity-based functionalities. We classify these interactions in two major categories:

Beacon SDK ↔ Beacon SDK Interactions. We find 17 beacon SDKs invoking APIs from one another. Notably, AltBeacon and Gimbal provide APIs frequently used by other beacon SDKs to perform BLE scans and access location data. For example, X-Mode, a location data aggregator, uses AltBeacon's APIs for BLE scan results, while mParticle, a data integration platform, calls Radar's APIs for location tracking. In some cases, multiple beacon SDKs coexist within the same app as in the French versions of McDonald's and Burger King apps to locate user at their premises, both using AltBeacon for beacon scanning and indoor location tracking, and

[8]These percentages are relative to the total number of applications found with at least one beacon SDK ($N = 9,976$).

Woosmap for geofencing. Woosmap also invokes APIs from third-party SDKs, such as Urban Airship, Salesforce Marketing Cloud, and Batch, via REST/OAuth APIs, enabling seamless data sharing and SDK-to-SDK integration.

Beacon SDK ↔ ATS Interaction. We identify 24 beacon SDKs sharing BLE/WiFi scan results and geolocation data with 21 non-beacon ATS SDKs. This behavior is concerning as advertising SDKs like CleverTap and Twitter MoPub can use this information for secondary purposes. Specifically, 11 ATS SDKs (*e.g.,* UrbanAirship) invoke WiFi scanning APIs of 6 different beacon SDKs (*e.g.,* Gimbal), and 16 ATSes invoke geolocation APIs from 22 beacon SDKs. Interestingly, the Yinzcam Sobek SDK calls APIs from Fluzo, an Automatic Content Recognition (ACR) SDK, for audio fingerprinting in the La Liga app (version 7.2.1), a potential GDPR violation that was investigated by the Spanish Data Protection Agency (AEPD) [43]. Such practices highlight the complexity of data sharing and synchronization across-SDKs for secondary purposes like advertising.

Unfortunately, these capabilities introduce unexpected security and privacy concerns due to Android's coarse-grained sandbox and permission model. Since there is no security boundary between libraries within the same app process, one library can interact with another (*e.g.,* invoking functions) to access IDs and sensitive data without restrictions or user awareness [124]. Unlike the same-origin policy for limiting cross-site tracking on the web, Android's lack of cross-SDK isolation mechanism allows them to access and share data within the same app process without any restrictions.

6 Privacy Analysis

Most of the analyzed apps with beacon SDKs (77%) self-disclose the collection of user data through Google Play's data safety labels. The most frequently collected data are device IDs (62%), personal information such as names, emails, and other user IDs (60%), and location (35%) supposedly for app functionality (69%) and analytics (68%), followed by account management (51%), and advertising or marketing (36%). However, this information is self-disclosed by app developers and may be erroneous or intentionally deceptive.

In this section, we use our dynamic analysis pipeline (§4.4) to analyze the runtime data collection practices associated with beacon SDKs across our 9,652 apps. Specifically, we study their permission requests (§6.1) and monitor the dissemination of 18 sensitive data types including beacon-specific data and user IDs (§6.2), demonstrating how beacon data is often bridged with device IDs like the AAID (§6.3). To contextualize our permission analysis in §6.1, we conclude with an evaluation of how developers follow permission consent best practices (§6.4). The results presented in this section constitute actual evidence of the dissemination of geolocation data linked to user IDs from apps to cloud services and across SDKs, for potential secondary purposes like identity profiling, mobility tracking, advertising and potentially data brokerage.

6.1 Permission Analysis

We compare declared permissions in each app's manifest with statically identified APIs to flag cases of over-permissioning. Our analysis shows that 82% of the apps request either fine or coarse-grained location permissions, 79% of beacon-enabled apps request WiFi permissions and 62% request Bluetooth-related permissions. This suggests the existence of close ties between beacon SDKs and geolocation services. Interestingly, 28% of these apps request audio permissions, which in some cases could be used for ultrasonic beaconing [9, 86, 97]. However, the analysis of audio-tracking technologies is outside the scope of this paper. Finally, over 40% of apps request phone state permissions to access the IMEI on Android 9 and 35% the AAID on Android 12.

Table 3: Usage of BLE, Location, and WiFi permissions by the top 15 SDKs, showing manifest declarations, API calls, and specific BLE_SCAN tags introduced in Android 12+.

SDK Name	# apps	# Android 12+	BLE		Location		WiFi		BLE_SCAN	
			Man.	API	Man.	API	Man.	API	% Req	% Tag
AltBeacon	4024	1822	95%	91%	96%	0%	73%	0%	57%	18%
Adobe	1328	514	35%	0%	80%	0%	89%	0%	9%	25%
Kochava	1118	499	20%	60%	42%	62%	97%	66%	5%	31%
Salesforce	1080	713	33%	0%	80%	0%	72%	0%	12%	25%
Estimote	510	197	97%	93%	88%	58%	53%	0%	17%	18%
LeanPlum	456	231	18%	0%	60%	0%	98%	74%	13%	14%
Gimbal	396	160	98%	82%	99%	81%	99%	58%	24%	3%
Radius Networks	369	94	86%	83%	91%	0%	84%	0%	31%	21%
Ad4Screen	198	75	0%	0%	0%	0%	0%	0%	8%	17%
Kontakt	195	82	94%	91%	99%	0%	62%	0%	84%	3%
Swrve	153	76	14%	0%	42%	0%	86%	0%	5%	25%
Exponea	99	68	24%	3%	78%	3%	79%	3%	6%	50%
Radar	93	61	38%	32%	99%	0%	80%	0%	25%	7%
Bazaarvoice	88	67	38%	0%	95%	0%	88%	0%	19%	31%
Rover SDK	50	26	98%	70%	94%	0%	84%	0%	12%	67%
Zendrive	34	16	85%	59%	100%	0%	100%	79%	75%	8.30%
Sensoro	4	2	100%	75%	100%	75%	75%	0%	100%	50%

Abbreviations Used: # Android 12+ - Number of apps targeting Android 12 or above; % Req - Percentage of BLE_SCAN permission requests; Man. - Manifest; API - API calls; % Tag - Percentage of apps using the neverForLocation tag.

We inspect the code of apps requesting these permissions to determine whether the caller class belongs to the app itself (first-party code) or the beacon SDK (third-party code) using the backward slicing technique described in §4. This process allows us to infer the usage and purpose of API invocations by reasoning about the business models of the SDKs requesting them according to publicly available information. In line with our static permission analysis, we find that API usage for location tracking closely matches declared permissions. For WiFi, we observe an 11% decrease in API usage compared to declared permissions, suggesting possible over-permissioning or the presence of obfuscated code. For Bluetooth, 18% of apps invoke Bluetooth APIs without declaring the corresponding permissions. This under-permissioning could stem from cases like Facebook and Ogury Ad SDKs, which provide multiple functionalities (i.e., advertising, location services, BLE), while app developers may use them just for very specific purposes.

Meanwhile SDKs like, AltBeacon and Estimote seem to have legitimate high rates of Bluetooth-related permission requests—95% and 97%, respectively—for detecting and interacting with beacons. Interestingly, 91% of apps integrating AltBeacon and also requesting location permission never invoke the corresponding AltBeacon location APIs in Java code, indicating that the permission request could be a requirement for complying with Android versions 11 and lower. On the contrary, we observe how the majority of apps integrating Estimote, LeanPlum, Gimbal or Radius Networks make use of the corresponding location APIs, as described in §5.

The dual purpose of many Android permissions has been criticized for causing confusion among developers and users. To illustrate this, we analyze the manifest files to extract the presence

Table 4: SDKs collect location data and exfiltrate identifiers classified under global persistent IDs, app persistent IDs, global resettable IDs, app resettable IDs, WiFi/Bluetooth scans, and GPS data (§3). Symbols indicate data collection on Android 9 (◇), Android 12 (▼), or both (★). Highlighted rows represent SDKs from our dataset; others are co-embedded SDKs detected through dynamic analysis.

SDK (# of Apps)		Glob. Pers.						App. Pers.	Glob. Rst.		App Rst.	WiFi/BLE scan						GPS	
	# App. Installs	Boot ID	GSF ID	IMEI	HW ID	WiFi MAC	Email	Android ID	AAID	BLE Name	FID	BLE ibeacon MAC	ibeacon UUID	Router MAC	Router scan MAC	Router Scan SSID	Router SSID	Coarse gelog.	Fine geoloc.
Kochava (220)	2B							◇	★	★				★			★		
Yandex (220)	572M			◇		★		◇	★					★	★	★	★		
Amplitude (190)	988M			◇			◇	◇	★	★								★	★
Datadog (140)	146M			◇				◇	★		★			★			★	★	★
Sentry (81)	53M					◇		◇	★	◇	▼			★			★	★	★
Omniture (49)	1B					◇		◇	★								★	★	★
Forter (34)	297M			◇	◇			◇		◇						★	★		
Radar (33)	280M							◇	◇				★					★	★
Huq Sourcekit (24)	25M							◇	★	★	★			◇			◇	★	★
cellrebel (18)	134M													★			★	★	★
Vizbee (16)	164M							◇	★					★			★	★	★
Cuebiq (6)	50M								★					★	★	★	★	◇	◇
taobao (6)	1B			◇				◇						★					
My Tracker SDK (6)	132M								★					★			★		
AdsWizz (6)	9M								★								◇	★	★
phunware (5)	78K								★		★			◇			★	★	★
conviva (5)	134M							◇	★					★			★	★	★
PayPal (5)	100M		★	◇	◇			◇						★	◇		★	★	★
Singlespot (5)	30M								★			★							
Incognia (5)	38M	★						◇	★					★	★	★	★	★	★
Colocator (4)	395K							◇						★					
Swrve (3)	9M							◇	★										
JPush (3)	346K															★	★		
Kontakt (3)	23K											★							
Proxy Cloud (2)	7M													★	★	★	★	★	★
pingID (1)	3M					◇		◇								▼	★		
appICE (1)	3M								★					★			★	◇	◇
Tangerine (1)	3M								★								★		
Proximi.io (1)	162K												▼						

of the `BLUETOOTH_SCAN` permission and the usage of the *neverForLocation* flag, introduced for apps targeting Android 12 or higher, as the official Android documentation recommends [34]. As summarized in Table 3, only 18% of apps that include AltBeacon and request Bluetooth scan permissions set also the *neverForLocation* flag, indicating that the remaining apps likely use this permission both for BLE scanning and location purposes. Furthermore, only 3% of the apps integrating Kontakt SDK set this flag. Among the SDKs offering location data aggregation—*e.g.,* Kochava, Adobe Experience Platform, and Salesforce Marketing Cloud—the use of this flag ranges from 25% to 35%. We observe that, while apps can use the *neverForLocation* flag with the `NEARBY_WIFI_SCAN` permission, only 11 apps request this permission, and only 6 set the flag.

Overall, our results show that only a minority of apps using scan APIs explicitly declare the intention of not to use these methods for location tracking purposes through mechanisms like the *neverForLocation* flag. Meanwhile, a significant proportion of apps integrating SDKs like Gimbal, LeanPlum, or Kochava appear over-permissioned, requesting more access than may be necessary. Unfortunately, these practices can create potential opportunities for third-party data harvesting, as we show next.

6.2 Beacon Data Collection

We study the runtime data access practices of beacon-enabled apps using the dynamic analysis pipeline described in §4.4. We find that they collectively contact over 25K unique domains, with 86% of apps and 20% of domains collecting at least one of the sensitive data types detailed in Table 4. While our dynamic analysis results are a lower-bound estimation of the potential practices due to inherent coverage limitations of runtime analysis, they provide actual evidence of SDK behaviors and potential privacy abuses. Additionally, dynamic analysis extends SDK detection beyond static methods by uncovering wireless scanning behaviors in advertising, analytics, and fraud-detection SDKs missed by static analysis due to code obfuscation or reflection (§4.5).

We observe discrepancies between the app's runtime behavior and the declared safety labels. We find that 2,292 apps collect device identifiers without disclosing them. For location data, a different pattern emerges: only 23% of the 3,535 apps both declare the collection of location data and transmit it at runtime while 563 apps transmit location data without disclosing it in their data safety labels. For the rest of the analysis, we report how SDKs collect beacon and geolocation data across apps and distinguish beacon SDKs from non-beacon SDKs that also exhibit beacon scanning behaviors.

Router: SSIDs and BSSIDs represent the names and MAC addresses of connected and nearby WiFi networks. They serve as precise location proxies [105]. We find that 1.95% and 1.62% of apps collect the router SSID and BSSID of their connected WiFi access points, respectively. At the SDK level, 29 of them collect either router SSIDs or BSSIDs, with only 4 SDKs collecting both. Furthermore, 0.32% of the apps and 8 different SDKs extend their reach to the SSID and BSSID of nearby WiFi access points. Leading beacon SDKs collecting AP data include Kochava, Colocator, Cuebiq—which was removed from the Play Store due to invasive data collection practices [69]—and non-beacon SDKs like Yandex, JPush [106], and Incognia that offer advertising, fraud prevention, and push notifications solutions [66]. Jointly, these SDKs account for 4B total installs.

BLE: we observe 218 apps disseminating the device's Bluetooth name—*i.e.,* the user-friendly name often defined by users and potentially containing PII like a user's name [31]—to cloud services. Additionally, 18 apps collect the AltBeacon UUID and 6 collect the iBeacon UUID, both of them being unique and persistent IDs tied to a specific physical location. At the SDK level, 7 beacon SDKs collect the device's Bluetooth address, 3 targets the iBeacon UUID, and 1 collects the iBeacon MAC address. Huq Sourcekit, Kontakt, and Radar are the most prevalent SDK with such capabilities, with a cumulative install count of 280M users. These results suggest that BLE scanning is comparatively less prevalent than WiFi scanning. This discrepancy could stem from the more controlled scanning methods offered by SDKs, or the use of geofencing—where scans are only enabled within specific areas—that limit our dynamic testing approach, as detailed in §4.5.

GPS Sensors: 20% and 18% of apps actively collect coarse or fine geolocation data, respectively. The majority of apps with beacon SDKs declare access to Android geolocation permissions, either fine (78%) or coarse location (77%) and even when the app is put in the background (20%). The 13 beacon SDKs collecting location data

```
com.geomobile.tiendeo outbound to api.radar.io:443
POST /v1/logs HTTP/1.1
Content-Type: application/json
Host: api.radar.io
..
"androidid":"XXXXXXXXXXXXXX"
{
    "createdAt": 1720523270332,
    "level": "DEBUG",
    "message": "Ranged beacon | beacon.type = IBEACON; beacon.uuid = 01022022-
          fa0f-0100-00ac-dd1c6502da1c; beacon.major = 53479; beacon.minor =
          42571; beacon.rssi = -12"
},
{
    "createdAt": 1720523270334,
    "level": "DEBUG",
    "message": "Handling beacon entry | beacon.type = IBEACON; beacon.uuid =
          01022022-fa0f-0100-00ac-dd1c6502da1c; beacon.major = 53479; beacon.
          minor = 42571; beacon.rssi = -12"
},

"events":[], "nearbyGeofences":[{"_id":"6318a0381c18820019e1e07e", "live":true,
"type":"circle", "tag":"es.s.m","externalId": "103769","geometryCenter":
{"coordinates":[longtitude: -X.XXXXX,latitude: XX.XXX],"type":"Point"},
```

Figure 4: iBeacon advertisements and geofence data exfiltrated to Radar.io, including Android ID, beacon details (UUID, major, minor, RSSI), and geofence metadata (coordinates and type).

include Radar, Salesforce Marketing Cloud, Rover SDK, LeanPlum, and Huq Sourcekit, with a cumulative install count of 4B devices. Additionally, we observe 33 non-beacon SDKs within the same apps accessing the location data, with OneSignal, Amplitude, Braze, and Flurry being the most prevalent ones. Among the SDKs that collect the device's GPS location, 15 gather also the WiFi AP SSID and BSSID. Similarly, as shown in Figure 4, Radar SDK collects the device's GPS location along with BLE ibeacon data to potentially improve location accuracy and enable precise location tracking [16]. Yet, this collection of location data solely for advertising or analytics purposes may potentially violate Google's Play Store policies [98].

Side-channels: we identify two non-beacon SDKs exploiting known vulnerabilities in older Android versions to perform wireless scans by bypassing permission requirements as of August 2024. The presence of such SDKs on the Play Store suggests that Google Play Protect may fail to detect apps invoking vulnerable APIs.

- **Vizbee SDK** is a third-party library for mobile-to-TV deep-linking present in apps with over 164M installs. Vizbee exploits the side channel vulnerability CVE-2020-0454 [95] to collect AP's SSIDs in Android versions 9 and below. The SDK uses the *onCapabilitiesChanged* callback function, registered via *ConnectivityManager's NetworkCallback*, which inadvertently provides SSID data. Vizbee caches and transmits this data to `metrics.clasptws.tv` under the key `WIFI_SSID` (See Table 12 in the Appendix). The transmission also includes `GEO_LAT` and `GEO_LONG` fields with values set to UNKNOWN, indicating that location permission is denied in this test. Vizbee stores the SSID in a variable named *hackedSsid*, suggesting a deliberate attempt to bypass Android permission controls.
- **Forter SDK** (v2.4.11) present in apps with over 297M installs, exploits the legacy API *'WifiConfiguration.SSID'* in Android 9 devices to collect SSIDs without requiring location permissions. According to current market shares, these vulnerabilities remain active on approximately 9% of Android devices globally [116].

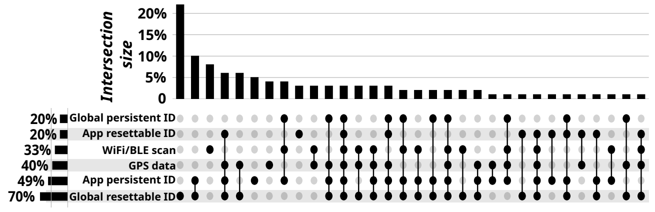

Figure 5: The UpSet plot illustrates how SDKs collect different combinations of ID categories. The top bars represent the percentage of SDKs collecting specific combinations (indicated by connected dots below), while the left bars show the total percentage of SDKs collecting each category, regardless of other data types collected.

This SDK transmits sensitive data, including SSIDs, security types (*e.g.,* WPA_PSK), network associations, and device IDs like the IMEI. In contrast, version 2.4.12 of the Forter SDK integrated in apps targeting Android 12 collects even more extensive network data, from DNS and DHCP details to WiFi scan results of nearby SSIDs and their security configurations, device traffic statistics, device IDs, and DNS settings.

6.3 ID Bridging

ID bridging is a privacy-intrusive practice that weakens the privacy protections of resettable identifiers (*e.g.,* AAID) and allows the creation of rich and persistent user profiles for advertising, identity profiling, and surveillance. We identify 61 SDKs potentially performing ID bridging, including 16 beacon SDKs and 45 non-beacon SDKs. Figure 5 summarizes the instances of ID bridging captured by our dynamic analysis pipeline, showing how beacon SDKs bridge various types of device IDs, geolocation, and BLE/WiFi scan results.

Overall, 70% of the beacon SDKs also collect global resettable IDs like the AAID and BLE device name, and 41% collect global persistent IDs like the GSF ID. Oftentimes, both persistent and resettable global IDs are bridged alongside WiFi or BLE scans (32% of all SDKs) or geolocation coordinates directly collected from the GPS sensor (39% of SDKs). We note that the uniqueness of human mobility traces makes user de-anonymization through geolocation data feasible, as demonstrated in prior work [27]. ID bridging strengthens this risk by linking mobility data with other identifiers, allowing precise user re-identification and persistent tracking. Prominent beacon SDKs such as Kochava, Adobe Experience Platform, and Cuebiq, as well as non-beacon SDKs present in the analyzed apps like Yandex, Adjust and AdColony appear to perform these practices. We highlight some notable cases next.

WiFi/BLE Scans with Global IDs: 33% of SDKs collect WiFi or Bluetooth scan data, 10% combine this data with global persistent IDs, and another 15% with resettable IDs. These practices typically follow three main patterns:

- **Connected Router Data:** 19% of SDKs collect either resettable or persistent user/device IDs along with connected WiFi network information (*e.g.,* router MAC, router SSID), while 15% of SDKs extend this by linking the Android ID with connected router details. Namely, beacon SDKs such as Kochava, Adobe Experience

Table 5: Packages with Yandex SDK, collecting multiple IDs.

						Router			
Package Name	**Installs**	**AAID**	**Android ID**	**WiFi MAC**	**IMEI**	**MAC**	**SSID**	**Scan MAC**	**Scan SSID**
ru.dvaberega	968K	x				x	x		
com.numplates.nomera3	576K	x				x	x	x	x
com.zenhotels.android	261K	x	x	x	x	x	x	x	x
com.wromanticgirlgame_10378860	60K	x				x	x	x	x
com.weSPPTPBBKotaBogor_13752947	39.4K	x				x	x	x	x
com.ratehawk.android	34.5K	x	x	x	x	x	x	x	x
com.wMadaniQaidahUrdu_15187981	515	x				x	x	x	x
com.wBukuMenwa_15656972	179	x				x	x	x	x
com.AirportTransportation_4990860	85	x				x	x	x	x

Platform, and Colocator, along with non-beacon SDKs like Conviva, PingID, AppICE, Taobao, Vizbee, and Datadog, combine the AAID with the persistent GSF ID. For example, Figure 7 shows Kochava collecting AAID with connected WiFi MAC. Similarly, non-beacon SDKs like Yandex and Alipay incorporates the Android ID and WiFi MAC, enabling cross-session tracking linked to precise network settings, regardless of AAID resets. We observe these practices in 5% of the apps that we analyze.

- **Nearby WiFi Scans:** Six SDKs upload nearby WiFi network data (*e.g.,* router scan SSID, router scan MAC), along with user IDs. For instance, Cuebiq and Incognia combine AAID and Android ID alongside both nearby and connected router scan data to track user movements across WiFi contexts, as shown in Figure 8. This data may be collected to enhance location services via crowdsourcing. Notably, Yandex combines AAID, Android ID, WiFi MAC, IMEI (in Android 9 and below), and even clipboard data with both connected and nearby router data. Table 5 lists popular apps with this behavior integrating the Yandex SDK.
- **BLE Beacon Data:** our analysis shows that three SDKs collect BLE beacon IDs such as iBeacon UUIDs or MAC addresses. This approach offers finer granularity than WiFi-based data and GPS location indoors. Proximi.io collects iBeacon UUIDs with device fingerprints, while the Kontakt SDK gathers iBeacon MAC addresses. Radar further integrates iBeacon UUIDs with GPS data, Android IDs, and AAIDs to enable detailed location-based tracking tied to physical BLE infrastructure even after AAID resets.

GPS Data with Global Resettable or Persistent IDs: 39% of SDKs collect GPS data along with user IDs. This includes beacon SDKs such as LeanPlum, Huq Sourcekit, and Cuebiq, and non-beacon SDKs like Vizbee, Incognia, and Conviva. Interestingly, Incognia goes a step further by harvesting the Boot ID, a 64-bit unique hex string ID generated on a device's first boot and stored securely, that should remain constant for the lifetime of the device unless the user performs a factory reset.

AAID Bridging: We find that 14% of SDKs link the AAID with global persistent IDs such as the IMEI (available only on Android 9 or below) or GSF ID. In more than 180 apps, beacon-enabled analytics and marketing SDKs such as Adobe Experience Platform and LeanPlum appear alongside non-beacon counterparts like Amplitude, Sentry, and MixPanel, which appear to bridge the IMEI with the AAID. To protect users' privacy, the Google Play Store user data policy explicitly prohibits linking persistent IDs with resettable ones like AAID for advertising or analytics purposes unless explicitly disclosed to users via privacy policies or in-app consent dialogs [38]. However, our findings suggests that such linking still occurs, which could potentially enable cross-app tracking, profiling, targeted advertising, and third-party data aggregation for resale. We also find non-beacon SDKs like MixPanel, Amplitude, and New Relic bridging other IDs like the Android ID with the IMEI (in Android 9 and below) in 244 of the analyzed apps. MixPanel further enriches this profile by adding GSF ID. The privacy implications of our findings are concerning. In fact, beacon SDKs like Huq Sourcekit, Radar, Incognia, and Vizbee seem to opportunistically attempt to piggyback on the permissions requested by the app developer to collect the full spectrum of IDs along with GPS and BLE/WiFi data. For geolocation data, SDKs like Huq Sourcekit and Radar do not directly invoke location APIs but instead appear to collect location data indirectly via host app permissions. We demonstrate the risks of ID bridging with two case studies:

- **Amplitude:** This mobile analytics platform performs extensive data collection practices by bridging precise GPS coordinates with IDs like the AAID, the Android IDs, and the IMEI (in Android 9 and below). Combining location data with persistent or resettable IDs for analytics purposes may violate Google's policy [98]. Additionally, Amplitude collects user email addresses, as seen in apps like `br.com.brainweb.ifood` (100M+ downloads), where emails are transmitted during the *Sign In with Google* OAuth event triggered in our testing. Of particular concern is Amplitude's ability to track every user interaction within the app that embeds the SDK. Each user-triggered event is transmitted to Amplitude's servers, along with precise GPS coordinates, hence potentially enabling semi-continuous tracking of users' interactions and the exact locations where they occur.
- **Adobe Experience Platform** collects various user identifiers through *demdex.net* and **.omtrdc.net* domains, including resettable (AAIDs), globally persistent (WiFi MACs, hashed emails), geolocation (router SSIDs, GPS), and app- or device-scoped IDs (Firebase installation IDs, Android IDs). Additionally, Adobe sets its proprietary *marketingCloudId*, a *"persistent and universal identifier designed to track users across all Adobe Experience Cloud products and subsidiaries"* [2]. Such SDK-specific IDs add opacity, limit user control and enabling potential ID bridging. In over 16 apps, we find the *adobedtm* tracker to collect hashed emails alongside the *marketingCloudId.* This data is stored in a dictionary labelled "PII" and logged under an event named *PiiInformationReceived*, suggesting an intentional collection of PII. We also observe cross-library interactions and potential data sharing. In `com.totalwine.app.store` (1M+ downloads), the *marketingCloudId* is shared with Appsflyer alongside AAIDs, suggesting a direct interconnection with Adobe's identity graph solutions.

6.4 Permission Usage Rationale

Android's official developer guidelines recommend that apps clearly communicate the necessity of the permission requests and the potential impact if the user denies it [33]. Android 6.0 introduced the *shouldShowRequestPermissionRationale()* method to inform developers if their app should explain to the user the purpose of a requested dangerous permission.

Beacon SDK-enabled apps pose significant privacy risks to users while offering them little to no control over data collection. To demonstrate this, we measure their compliance with Android's permission request guidelines when requesting dangerous permissions. We do this by statically searching for instances where beacon SDKs invoke the *shouldShowRequestPermissionRationale()* and *requestPermissions()* methods. We then parse the permission strings associated with these function calls. Finally, we analyze the app's UI components to detect any consent dialogs, such as alerts and banners, that might be used to explain the permissions. By contextually linking all these elements, we infer whether the app clearly provides a rationale for the requested permissions.

We find that 24% of apps in our dataset do not implement the *shouldShowRequestPermissionRationale()* API and include a rationale for permission requests. Among those that do implement the API, 92% rely on third-party SDKs to provide such explanations correctly, a transparency feature beyond the control of app developers. Additionally, the presence of these justifications varies significantly across permission types: 71% of apps requesting location data fail to justify access, only 13% of apps justify *FINE_LOCATION* access, 7% for *COARSE_LOCATION*, and 8% for *BACKGROUND_LOCATION*, in all cases primarily handled by third-party SDKs. The lack of justification is even worse for Bluetooth permissions, with 97% of apps providing no explanation despite its known tracking risks. Our results suggest that, most developers underestimate BLE data risks compared to location data.

On the other hand, only five beacon SDKs in our dataset use the *shouldShowRequestPermissionRationale()* API to explain specific dangerous permissions. Estimote shows rationales for coarse location in 75% of apps embedding it. Radar, Salesforce Marketing Cloud, and Singlespot show fine location rationales in 42%, 54%, and 55% of apps, respectively. Additionally, InMarket provides rationales for accessing background location in 52% of apps. In contrast, SDKs like Swrve and IndoorAtlas delegate this responsibility entirely to app developers despite invoking permission-protected APIs themselves. Swrve implements a callback to check if the rationale was shown to the user [118], allowing it to piggyback on the app's permissions.

These findings highlight systemic transparency failures in how beacon-enabled apps handle permission requests, leaving users with little insight or control despite the dual usage of these permissions. Stricter auditing, policy interventions and broader ecosystem reforms are necessary to address these gaps as we discuss next.

7 Discussion

Our study reveals the pervasive tracking risks of beacon-enabled SDKs in Android. By analyzing behaviors, cross-library interactions, and data collection practices, we show how beacon SDKs track users across apps and services. Building on Dehaye and Reardon's premise [29], our findings challenge the assumption that BLE-based distance authentication is secure against global passive adversaries. The belief that large-scale Bluetooth surveillance is economically unfeasible is flawed, as SDKs turn millions of everyday apps into passive scanners. This shifts costs—power and data collection—to users while enabling tracking of individuals who never installed these apps. Their wireless devices, such as headphones or AirTags, can be continuously observed by passive scanners. While our focus is on mobile apps, similar tracking risks may exist—including connected platforms (*e.g.,* IoT)—highlighting the need for better transparency, accountability, and regulatory safeguards.

Beacon SDKs and Location Surveillance. In §6 we demonstrate how beacon SDKs continuously scan for WiFi, Bluetooth, and GPS signals, supporting a vast data ecosystem of targeted advertising and data brokers [73]. For instance, Singlespot, a French marketing firm, claims to have collected data from 2 million users and sells it for up to $20,000 via platforms like Datarade [113]. If persistent identifiers are linked to specific geolocations, SDKs can then leverage WPS, data brokers and public databases to infer the users' location and accurately track their movements. Such knowledge is harmful to users' privacy because it can disclose information about their personal beliefs or sexual orientation, or it can even reveal classified information such as the location of secret facilities [27, 42, 59, 128]. Beyond first-hand data collection, beacon SDKs may also share or repurpose the data they obtained [73]. As we analyzed in §5.2, SDKs co-located on the same app can exchange the collected information. This interconnectivity enables pervasive data flows across organizations, where user-tagged beacon data spreads across multiple third parties. Similarly, as shown in §6.3, proprietary identifiers like Adobe's Marketing Cloud ID get linked with persistent and resettable user IDs and then shared with other SDKs. These practices circumvent user control and blur accountability.

Platform Policy Enforcement. Google Play has introduced policies designed to protect user privacy by imposing strict rules on what data apps can collect and how they use it. According to these policies, developers are responsible for ensuring that any SDKs embedded in their apps do not sell or misuse sensitive user information they collect [99]. For instance, one policy explicitly prohibits linking user data or resettable IDs (e.g., AAID) with persistent device IDs or PII for advertising [98]. However, while analyzing apps in practice, we find that 14% of SDKs (§ 6.3) bridge resettable identifiers with persistent ones. Such identifier bridging undermines Android's privacy safeguards, as resetting identifiers no longer limits continuous user tracking. A second recital mandates clear user disclosure for any location data collected for ads, ensuring transparency and data minimization. Unfortunately, these policies shift the burden to developers. As they rush their market release, regulatory complexity, inexperience, and incomplete SDK documentation hinders compliance. To enforce compliance and detect policy violations, marketplaces could implement stricter independent audits before app release, runtime monitoring, and publicly disclose audit results for transparency.

Regulation and Transparency. Regulatory efforts in the EU and the US aim to mitigate privacy risks and enhance transparency in mobile platforms. GDPR [45] mandates data minimization and consent, yet enforcement remains a challenge due to opaque SDK data-sharing practices. Meanwhile, CNIL and AEPD have issued guidelines to curb passive wireless tracking [26, 28]. CNIL advises app developers to map data flows, SDK providers to document compliance, and marketplaces to enforce vetting. However, enforcement gaps persist as widespread over-permissioning (§6.1), proprietary identifier use by SDKs (§6.3), and uncontrolled SDK-driven data sharing beyond user control enable large-scale tracking, making regulatory oversight difficult. Our empirical findings reveal how beacon SDKs evade existing protections through wireless scanning, cross-SDK interactions, and non-compliant tracking. By highlighting these risks, our analysis calls for strengthening audits, mandating SDK disclosures, and stricter enforcement to ensure compliance and curb unchecked data collection.

Defense Measures. Our work highlights systemic transparency failures and policy gaps in the data collection practices of beacon-enabled apps and Android's transparency features. Mitigating these risks requires stronger privilege separation and sandboxing to limit cross-library data sharing. While Google's Privacy Sandbox is in the right direction, it only isolates advertising SDKs [56]. Stricter runtime audits and app store vetting are needed to detect covert SDK behavior, yet these rely solely on platform enforcement, leaving users with no control over beacon data collection and sharing. This lack of control is worsened by the absence of transparency mechanisms in the beacon ecosystem. As shown in §6.4, the *shouldShowRequestPermissionRationale()* API designed to improve user awareness is rarely used, exposing gaps in permission governance. Existing privacy controls offer little protection against abuse. The FCC suggests setting Bluetooth to hidden mode, but this only prevents device discovery, not beacon scanning [17]. Addressing these risks requires a multi-layered approach, including stricter app store policies, independent audits, and technical safeguards to curb unregulated beacon tracking. While our work focuses on policy and platform-level privacy controls, future research can explore usability improvements to enhance user awareness and transparency.

8 Related work

WiFi and BLE Scanning. Studies have shown how WiFi probe requests, which lack encryption and authentication, are exploited to track users and establish social links using SSIDs and BSSIDs [24, 50]. This data contributes to large databases created through techniques like wardriving, enabling passive location tracking by third parties [111]. Rye and Levin [110] highlighted the risks by revealing how Apple's WiFi Positioning System (WPS) mapped geolocations for over 2 billion BSSIDs globally, enabling mass surveillance. On the other hand, vulnerabilities in Bluetooth have been exploited to track users [14, 25, 48, 72, 79], spoof, and perform denial-of-service attacks [76] and exfiltrate private data [102]. Achara *et al.* [1] highlighted how Android's WiFi permissions were exploited by apps to infer user locations, prompting stricter controls by linking WiFi data to location permissions.

Several studies also identified privacy abuses across apps and SDKs. Reyes *et al.* [107] exposed children's apps harvesting WiFi data to infer locations, violating COPPA. Reardon *et al.* [105] uncovered covert channels in Android used to infer user locations. Dehaye and Reardon [29] showed SDKs like X-Mode harvesting Bluetooth scans for user tracking to demonstrate the existance of global passive adversaries in the context of contact tracing apps. Building on this, our work provides the first empirical, large-scale analysis of the Android beacon ecosystem, revealing how SDKs utilize GPS, BLE beacons, and WiFi signals to track user proximity with high precision and link this data to the user or device IDs.

Location. Location privacy has drawn significant research attention due to the rise of location-based services (LBS). A large body of work has focused on GPS data dissemination in location-based social networks (LBSNs) [40, 41, 135, 137, 138], crowdsourced location tracking [58, 112, 126, 133], and how GPS data from mobile apps [40, 41, 137] inadvertently reveals sensitive information such as military base locations or can be exploited for stalking and harassment [133, 138]. On the other hand, studies of Call Detail Records (CDRs) and Twitter geolocation data revealed the uniqueness and predictability of human mobility patterns [27, 42, 54]. De Montjoye *et al.* [27] demonstrated that with just four location points it is possible to uniquely identify 95% of individuals However, there is a gap in the literature on how location data is collected, who operates these location data services within the mobile ecosystem, and how they can be misused to track users and their movements.

Mobile App Privacy. Privacy risks in mobile apps have been extensively studied using static [10, 80, 87, 94, 125, 137] and dynamic [40, 78, 104, 105, 120] analysis techniques to uncover malicious behaviors, third-party code implications [49, 83, 104, 132], transparency issues [8, 46, 78], and regulatory compliance gaps [75, 77, 92, 93, 136]. However, the limitations of relying solely on static or dynamic analysis [15, 22, 52, 100] have led to hybrid approaches that more effectively expose side and covert channels [101, 105, 124] and privacy risks from analytical SDKs [30, 104, 115, 117]. Despite several proposed solutions like privacy policies and labels, discrepancies still persist between stated and actual data usage [8, 60, 132]. Our work does not use formal privacy accounting frameworks that rely on mathematical models to estimate privacy issues, such as data de-anonymization or re-identification [27, 40, 41, 133]. On the contrary, we align with prior empirical privacy research that combines static and dynamic analysis to measure the prevalence of beacon data-collection behaviors and estimate the number of affected users [49, 78, 81, 93, 104, 105, 107].

9 Conclusions

This paper presents the first large-scale empirical analysis of wireless-scanning SDKs in the Android ecosystem. By combining static and dynamic analysis with signal injection and runtime monitoring, we reveal how 52 SDKs across 9,976 apps exploit Bluetooth and WiFi scanning to infer user location and collect sensitive data. Our findings reveal that this ecosystem is tightly connected with advertising and tracking purposes and operates with minimal oversight. Most SDKs collect geolocation data for such secondary purposes and violate platform policies by engaging in ID bridging—linking persistent and resettable identifiers to construct detailed user profiles without user consent or knowledge for persistent user tracking. Some SDKs even intentionally exploit side channels to access sensitive data and IDs without requesting the pertinent Android permissions. We provide concrete evidence of non-compliance with platform policies and inadequate enforcement of existing rules. Our study demonstrates that existing privacy protections are insufficient, highlighting the need for stricter regulatory control, robust SDK sandboxing to limit cross-library data sharing, proactive audits to curb policy violations, and stronger transparency mechanisms to prevent large-scale tracking and ensure user control.

Acknowledgments

We thank the reviewers and the shepherd for their valuable feedback and suggestions for improving our paper. We used OpenAI's

ChatGPT to correct grammatical errors and to improve the clarity and coherence of our paper. The AI model was used with the following prompt: *"please improve the clarity and coherence of the text while maintaining the original intent"*. This research was partially supported by project PID2022-143304OB-I00 (PARASITE) funded by MCIN/AEI /10.13039/501100011033/ and by the ERDF “A way of making Europe”, the Spanish National Cybersecurity Institute (INCIBE) under Proyectos Estratégicos de Ciberseguridad – CIBERSEGURIDAD EINA UNIZAR, and by the Recovery, Transformation and Resilience Plan funds, financed by the European Union (Next Generation), the Natural Sciences and Engineering Research Council of Canada (NSERC) (funding reference number RGPIN/04237-2018), and the Spanish AEI grant CYCAD (PID2022-140126OB-I00). Dr. Srdjan Matic was partially funded by the Atracción de Talento grant (Ref. 2020-T2/TIC-20184), funded by Madrid regional government. Prof. N. Vallina-Rodriguez was appointed as 2019 Ramon y Cajal fellow (RYC2020-030316-I) funded by MCIN/AEI/10.13039/501100011033 and ESF Investing in your future. The opinions, findings, and conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of any of the funding bodies.

References

[1] Jagdish Prasad Achara, Mathieu Cunche, Vincent Roca, and Aurélien Francillon. 2014. Wifileaks: Underestimated privacy implications of the ACCESS_WIFI_STATE Android permission. In *Proceedings on Security and privacy in wireless and mobile networks*.

[2] Adobe. 2023. Identifying Visitors in Adobe Target Delivery API. https://experienceleague.adobe.com/docs/target-dev/developer/api/delivery-api/identifying-visitors.html.

[3] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *Proceedings of the 13th International Conference on Mining Software Repositories*.

[4] AltBeacon. 2024. AltBeacon Specification. https://github.com/AltBeacon/spec.

[5] Androguard. 2024. Androguard. https://github.com/androguard/androguard.

[6] Android Developers. 2025. UI/Application Exerciser Monkey. https://developer.android.com/studio/test/other-testing-tools/monkey

[7] Apple. 2024. Location Services and Privacy. https://support.apple.com/en-us/HT207056.

[8] Ioannis Arkalakis, Michalis Diamantaris, Serafeim Moustakas, Sotiris Ioannidis, Jason Polakis, and Panagiotis Ilia. 2024. Abandon All Hope Ye Who Enter Here: A Dynamic, Longitudinal Investigation of Android’s Data Safety Section. In *Proceedings of the USENIX Security Symposium*.

[9] Daniel Arp, Erwin Quiring, Christian Wressnegger, and Konrad Rieck. 2017. Privacy Threats through Ultrasonic Side Channels on Mobile Devices. In *IEEE European Symposium on Security and Privacy*.

[10] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. 2014. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In *ACM sigplan notices*.

[11] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, and David Lie. 2012. PScout: Analyzing the Android Permission Specification. In *Conference on Computer and Communications Security (CCS)*.

[12] Azira. 2024. Homepage. https://azira.com.

[13] Michael Backes, Sven Bugiel, Erik Derr, Patrick McDaniel, Damien Octeau, and Sebastian Weisgerber. 2016. On Demystifying the Android Application Framework: Re-Visiting Android Permission Specification Analysis. In *Proceedings of the USENIX Security Symposium*.

[14] Johannes Becker, David Li, and David Starobinski. 2019. Tracking Anonymized Bluetooth Devices. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*.

[15] Shauvik Roy Choudhary, Alessandra Gorla, and Alessandro Orso. 2015. Automated Test Input Generation for Android: Are We There Yet?. In *Proceedings on Automated Software Engineering (ASE)*.

[16] Wolfie Christl. 2024. Tracking Indoor Location, Movement and Desk Occupancy in the Workplace. https://crackedlabs.org/en/data-work/publications/indoortracking.

[17] Federal Communications Commission. 2024. How to Protect Yourself Online. https://www.fcc.gov/consumers/guides/how-protect-yourself-online.

[18] Federal Trade Commission. 2016. Mobile Advertising Network InMobi Settles FTC Charges It Tracked Hundreds of Millions of Consumers’ Locations Without Permission. https://www.ftc.gov/news-events/news/press-releases/2016/06/mobile-advertising-network-inmobi-settles-ftc-charges-it-tracked-hundreds-millions-consumers.

[19] Federal Trade Commission. 2024. FTC Order Prohibits Data Broker X-Mode/Outlogic from Selling Sensitive Location Data. https://www.ftc.gov/news-events/news/press-releases/2024/01/ftc-order-prohibits-data-broker-x-mode-social-outlogic-selling-sensitive-location-data. In *FTC Press Releases*.

[20] Federal Trade Commission. 2024. FTC v. Kochava Inc. https://www.ftc.gov/legal-library/browse/cases-proceedings/ftc-v-kochava-inc.

[21] Federal Trade Commission. 2024. How "Location, Location, Location" Can Lead to "Enforcement, Enforcement, Enforcement". https://www.ftc.gov/business-guidance/blog/2024/01/how-location-location-location-can-lead-enforcement-enforcement-enforcement. In *FTC Business Guidance Blog*.

[22] Andrea Continella, Yanick Fratantonio, Martina Lindorfer, Alessandro Puccetti, Ali Zand, Christopher Kruegel, Giovanni Vigna, et al. 2017. Obfuscation-Resilient Privacy Leak Detection for Mobile Apps Through Differential Analysis.. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[23] Cue. 2024. Cue - Creating Unforgettable Experiences. https://www.connectwithcue.com/.

[24] Mathieu Cunche, Mohamed Ali Kaafar, and Roksana Boreli. 2012. I know who you will meet this evening! linking wireless devices using wi-fi probe requests. In *IEEE Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE.

[25] Aveek K. Das, Parth H. Pathak, Chen-Nee Chuah, and Prasant Mohapatra. 2016. Uncovering Privacy Leakage in BLE Network Traffic of Wearable Fitness Trackers. In *International Workshop on Mobile Computing Systems and Applications*.

[26] Commission Nationale de l’Informatique et des Libertés (CNIL). 2024. Recommandation relative aux applications mobiles. https://www.cnil.fr/sites/cnil/files/2024-09/recommandation-applications-mobiles.pdf.

[27] Yves-Alexandre de Montjoye, Cesar Hidalgo, Michel Verleysen, and Vincent Blondel. 2013. Unique in the Crowd: The privacy bounds of human mobility. In *Nature*.

[28] Agencia Española de Protección de Datos. 2024. Tecnologías de seguimiento Wi-Fi: Orientaciones para responsables del tratamiento. https://www.aepd.es/guias/orientaciones-wifi-tracking-seguimiento.pdf.

[29] Paul-Olivier Dehaye and Joel Reardon. 2020. Proximity Tracing in an Ecosystem of Surveillance Capitalism. In *Proceedings of the Workshop on Privacy in the Electronic Society*.

[30] Soteris Demetriou, Whitney Merrill, Wei Yang, Aston Zhang, and Carl A Gunter. 2016. Free for all! assessing user data exposure to advertising libraries on android.. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[31] Loleta Detweiler. 2023. How To Name Bluetooth Devices. https://robots.net/tech/how-to-name-bluetooth-devices.

[32] Android Developers. 2024. Android 6.0 Changes: Behavior Changes. https://developer.android.com/about/versions/marshmallow/android-6.0-changes#behavior-hardware-id.

[33] Android developers. 2024. App permissions best practices | Android Developers. https://developer.android.com/training/permissions/usage-notes.

[34] Android Developers. 2024. Bluetooth Permissions. https://developer.android.com/develop/connectivity/bluetooth/bt-permissions#assert-never-for-location.

[35] Android Developers. 2024. Find Bluetooth Devices. https://developer.android.com/develop/connectivity/bluetooth/find-bluetooth-devices.

[36] Android Developers. 2024. Permissions on Android. https://developer.android.com/guide/topics/permissions/overview.

[37] Android Developers. 2024. Request Location Permissions. https://developer.android.com/develop/sensors-and-location/location/permissions.

[38] Android Developers. 2024. User Data and Identifiers in Android. https://developer.android.com/identity/user-data-ids.

[39] Android Developers. 2024. Wi-Fi Scan. https://developer.android.com/develop/connectivity/wifi/wifi-scan.

[40] Karel Dhondt, Victor Le Pochat, Yana Dimova, Wouter Joosen, and Stijn Volckaert. 2024. Swipe Left for Identity Theft: An Analysis of User Data Privacy Risks on Location-based Dating Apps. In *Proceedings of the USENIX Security Symposium*.

[41] Karel Dhondt, Victor Le Pochat, Alexios Voulimeneas, Wouter Joosen, and Stijn Volckaert. 2022. A Run a Day Won’t Keep the Hacker Away: Inference Attacks on Endpoint Privacy Zones in Fitness Tracking Social Networks. In *Conference on Computer and Communications Security (CCS)*.

[42] Kostas Drakonakis, Panagiotis Ilia, Sotiris Ioannidis, and Jason Polakis. 2019. Please Forget Where I Was Last Summer: The Privacy Risks of Public Location (Meta)Data. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[43] Rachel England. 2018. Spanish soccer league app spied on fans to catch pirate broadcasts. https://www.engadget.com/2018-06-13-spanish-soccer-app-la-liga-spying-pirate-broadcast.html.
[44] Estimote. 2018. Estimote Proximity SDK for Android. https://github.com/Estimote/Android-Proximity-SDK.
[45] EU. 2018. General Data Protection Regulation. https://gdpr-info.eu/.
[46] Ming Fan, Jifei Shi, Yin Wang, Le Yu, Xicheng Zhang, Haijun Wang, Wuxia Jin, and Ting Liu. 2024. Giving without Notifying: Assessing Compliance of Data Transmission in Android Apps. In *Proceedings on Automated Software Engineering (ASE)*.
[47] Parvez Faruki, Ammar Bharmal, Vijay Laxmi, Manoj Singh Gaur, Mauro Conti, and Muttukrishnan Rajarajan. 2014. Evaluation of android anti-malware techniques against dalvik bytecode obfuscation. In *IEEE Conference on Trust, Security and Privacy in Computing and Communications*.
[48] Kassem Fawaz, Kyu-Han Kim, and Kang G. Shin. 2016. Protecting Privacy of BLE Device Users. In *Proceedings of the USENIX Security Symposium*.
[49] Álvaro Feal, Julien Gamba, Juan Tapiador, Primal Wijesekera, Joel Reardon, Serge Egelman, and Narseo Vallina-Rodriguez. 2021. Don't accept candy from strangers: An analysis of third-party mobile sdks. In *Data Protection and Privacy: Data Protection and Artificial Intelligence*.
[50] Julien Freudiger. 2015. How talkative is your mobile device? An experimental study of Wi-Fi probe requests. In *Proceedings on Security and Privacy in Wireless and Mobile Networks (WiSec)*.
[51] Guillaume Gagnon, Sébastien Gambs, and Mathieu Cunche. 2024. RSSI based attacks for identification of BLE devices. *Computers & Security*.
[52] Julien Gamba, Mohammed Rashed, Abbas Razaghpanah, Juan Tapiador, and Narseo Vallina-Rodriguez. 2020. An Analysis of Pre-installed Android Software. In *2020 IEEE Symposium on Security and Privacy (SP)*.
[53] Aniketh Girish, Tianrui Hu, Vijay Prakash, Daniel J. Dubois, Srdjan Matic, Danny Yuxing Huang, Serge Egelman, Joel Reardon, Juan Tapiador, David Choffnes, and Narseo Vallina-Rodriguez. 2023. In the Room Where It Happens: Characterizing Local Communication and Threats in Smart Homes. In *Proceedings of the Internet Measurement Conference (IMC)*.
[54] Marta C González, César A Hidalgo, and Albert-László Barabási. 2008. Understanding individual human mobility patterns. In *Nature*.
[55] Google. 2024. Geolocation API Overview. https://developers.google.com/maps/documentation/geolocation/overview.
[56] Google Developers. 2025. Privacy Sandbox on Android. https://developers.google.com/admob/android/privacy/sandbox
[57] Ben Greenstein, Ramakrishna Gummadi, Jeffrey Pang, Mike Y Chen, Tadayoshi Kohno, Srinivasan Seshan, and David Wetherall. 2007. Can Ferris Bueller Still Have His Day Off? Protecting Privacy in the Wireless Era. In *Proceedings of the USENIX Workshop on Hot Topics in Operating Systems (HotOS)*.
[58] Alexander Heinrich, Milan Stute, Tim Kornhuber, and Matthias Hollick. 2021. Who can find my devices? security and privacy of apple's crowd-sourced bluetooth location tracking system. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*.
[59] Urs Hengartner and Hui Zang. 2011. Anonymization of Location Data Does Not Work: A Large-Scale Measurement Study. In *Proceedings of the Conference on Mobile Computing and Networking (MobiCom)*.
[60] Hiroki Inayoshi, Shohei Kakei, and Shoichi Saito. 2024. Detection of Inconsistencies between Guidance Pages and Actual Data Collection of Third-party SDKs in Android Apps. In *Proceedings on Mobile Software Engineering and Systems*.
[61] Apple Inc. 2024. About Privacy and Location Services in iOS, iPadOS, and watchOS. https://support.apple.com/en-us/102515.
[62] Apple Inc. 2024. Exposure Notification Bluetooth Specification. https://developer.apple.com/documentation/exposurenotification.
[63] Apple Inc. 2024. Getting Started with iBeacon. https://developer.apple.com/ibeacon/.
[64] Google Inc. 2024. Eddystone Protocol Specification. https://github.com/google/eddystone.
[65] Google Inc. 2024. Exposure Notifications API. https://developers.google.com/android/exposure-notifications/exposure-notifications-api.
[66] Incognia. 2024. Incognia Documentation. https://developer.incognia.com/docs/.
[67] Infillion. 2007. Gimbal Ad Platform. https://infillion.com/gimbal-ad-platform-learn-more/.
[68] iTransition. 2024. Beacons in Retail: How They Work and Benefits for Businesses. https://www.itransition.com/retail/beacons.
[69] Adrianne Jeffries and Bennett Cyphers. 2020. How SDKs, hidden trackers in your phone, work. *Vox* (2020). https://www.vox.com/recode/2020/7/8/21311533/sdks-tracking-data-location
[70] The Wall Street Journal. 2024. House Investigating Company Selling Phone Location Data to Government Agencies. https://www.wsj.com/articles/house-investigating-company-selling-phone-location-data-to-government-agencies-11593026382.
[71] Tom Karpiniec. 2023. The Road to Good Bluetooth Permissions on Mobile. https://ditto.live/blog/bluetooth-permissions-on-mobile.
[72] Edden Kashi and Angeliki Zavou. 2020. Did I Agree to This? Silent Tracking Through Beacons. In *HCI for Cybersecurity, Privacy and Trust*.
[73] Jon Keegan and Alfred Ng. 2021. There's a Multibillion-Dollar Market for Your Phone's Location Data. https://themarkup.org/privacy/2021/09/30/theres-a-multibillion-dollar-market-for-your-phones-location-data
[74] Tom Kemp. 2022. A Look at the Different Types of Data Brokers. https://www.tomkemp.ai/blog/2022/10/05/a-look-at-the-different-types-of-data-brokers.
[75] Simon Koch, Benjamin Altpeter, and Martin Johns. 2023. The {OK} is not enough: A large scale study of consent dialogs in smartphone applications. In *Proceedings of the USENIX Security Symposium*.
[76] Constantinos Kolias, Lucas Copi, Fengwei Zhang, and Angelos Stavrou. 2017. Breaking BLE Beacons For Fun But Mostly Profit. In *Proceedings of the European Workshop on Systems Security*.
[77] Konrad Kollnig, Pierre Dewitte, Max Van Kleek, Ge Wang, Daniel Omeiza, Helena Webb, and Nigel Shadbolt. 2021. A fait accompli? an empirical study into the absence of consent to {Third-Party} tracking in android apps. In *Symposium on Usable Privacy and Security (SOUPS)*.
[78] Konrad Kollnig, Anastasia Shuba, Reuben Binns, Max Van Kleek, and Nigel Shadbolt. 2021. Are iphones really better for privacy? comparative study of ios and android apps. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*.
[79] Aleksandra Korolova and Vinod Sharma. 2018. Cross-App Tracking via Nearby Bluetooth Low Energy Devices. In *Proceedings of the ACM Conference on Data and Application Security and Privacy*.
[80] Haoran Lu, Qingchuan Zhao, Yongliang Chen, Xiaojing Liao, and Zhiqiang Lin. 2023. Detecting and measuring aggressive location harvesting in mobile apps via data-flow path embedding. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*.
[81] Allan Lyons, Julien Gamba, Austin Shawaga, Joel Reardon, Juan Tapiador, Serge Egelman, and Narseo Vallina-Rodríguez. 2023. Log:{It's} Big,{It's} Heavy,{It's} Filled with Personal Data! Measuring the Logging of Sensitive Information in the Android Ecosystem. In *Proceedings of the USENIX Security Symposium*.
[82] Ziang Ma, Haoyu Wang, Yao Guo, and Xiangqun Chen. 2016. LibRadar: fast and accurate detection of third-party libraries in Android apps. In *Proceedings of the International Conference on Software Engineering Companion*.
[83] Samin Yaseer Mahmud, K Virgil English, Seaver Thorn, William Enck, Adam Oest, and Muhammad Saad. 2022. Analysis of Payment Service Provider SDKs in Android. In *Proceedings on Computer Security Applications Conference*.
[84] Davide Maiorca, Davide Ariu, Igino Corona, Marco Aresu, and Giorgio Giacinto. 2015. Stealth attacks: An extended insight into the obfuscation effects on android malware. In *Computers & Security*.
[85] Célestin Matte, Jagdish Prasad Achara, and Mathieu Cunche. 2015. Device-to-identity linking attack using targeted wi-fi geolocation spoofing. In *Proceedings on Security and Privacy in Wireless and Mobile Networks (WiSec)*.
[86] Vasilios Mavroudis, Shuang Hao, Yanick Fratantonio, Federico Maggi, Christopher Kruegel, and Giovanni Vigna. 2017. On the Privacy and Security of the Ultrasound Ecosystem. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*.
[87] Yuhong Nan, Zhemin Yang, Xiaofeng Wang, Yuan Zhang, Donglai Zhu, and Min Yang. 2018. Finding clues for your secrets: semantics-driven, learning-based privacy discovery in mobile apps. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.
[88] Navizon. 2024. Navizon. http://www.navizon.com/.
[89] Neoma. 2024. IoT and Crowd Management: Insights and Solutions. https://neoma.ai/thinking/blog/13/iot-and-crowd-management.html.
[90] Radius Networks. 2014. Proximity Kit for Android. https://github.com/RadiusNetworks/proximitykit-android.
[91] The Hacker News. 2014. Spying Agencies Tracking Your Location. https://thehackernews.com/2014/01/spying-agencies-tracking-your-location_31.html.
[92] Trung Tin Nguyen, Michael Backes, Ninja Marnau, and Ben Stock. 2021. Share First, Ask Later (or Never?) Studying Violations of {GDPR's} Explicit Consent in Android Apps. In *Proceedings of the USENIX Security Symposium*.
[93] Trung Tin Nguyen, Michael Backes, and Ben Stock. 2022. Freely given consent? studying consent notice of third-party tracking and its violations of gdpr in android apps. In *Conference on Computer and Communications Security (CCS)*.
[94] Damien Octeau, Patrick McDaniel, Somesh Jha, Alexandre Bartel, Eric Bodden, Jacques Klein, and Yves Le Traon. 2013. Effective {Inter-Component} communication mapping in android: An essential step towards holistic security analysis. In *Proceedings of the USENIX Security Symposium*.
[95] National Institute of Standards and Technology. 2020. CVE-2020-0454 Detail. https://nvd.nist.gov/vuln/detail/CVE-2020-0454.
[96] Facundo Olano. 2015. Google Play Scraper. https://github.com/facundoolano/google-play-scraper.
[97] Moritz Pfister, Robert Michael, Max Boll, Cosima Kö rfer, Konrad Rieck, and Daniel Arp. 2024. Listening between the Bits: Privacy Leaks in Audio Fingerprints. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*.
[98] Google Play. 2023. Developer Program Policy. https://support.google.com/googleplay/android-developer/answer/9857753

[99] Google Play. 2024. User Data Policy for Android Developers. https://support.google.com/googleplay/android-developer/answer/10144311?sjid=18338717510598425318-EU.

[100] Sebastian Poeplau, Yanick Fratantonio, Antonio Bianchi, Christopher Kruegel, and Giovanni Vigna. 2014. Execute this! analyzing unsafe and malicious dynamic code loading in android applications.. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[101] Sajjad Pourali, Nayanamana Samarasinghe, and Mohammad Mannan. 2022. Hidden in plain sight: exploring encrypted channels in android apps. In *Conference on Computer and Communications Security (CCS)*.

[102] Joseph Priest and Daryl Johnson. 2015. Covert Channel over Apple iBeacon. In *Proceedings on Security and Management (SAM)*.

[103] Exodus Privacy. 2024. Homepage. https://exodus-privacy.eu.org/en/.

[104] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, Phillipa Gill, et al. 2018. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[105] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 2019. 50 ways to leak your data: An exploration of apps' circumvention of the android permissions system. In *Proceedings of the USENIX Security Symposium*.

[106] Joel Reardon, Nathan Good, Robert Richter, Narseo Vallina-Rodriguez, Serge Egel-man, and Quentin Palfrey. 2020. Jpush away your privacy: A case study of Jiguang's Android SDK. (2020).

[107] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, Serge Egelman, et al. 2018. "Won't somebody think of the children?" examining COPPA compliance at scale. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*.

[108] Ian Rose and Matt Welsh. 2010. Mapping the urban wireless landscape with Argos. In *Proceedings of the ACM conference on embedded networked sensor systems*.

[109] Bayerischer Rundfunk. 2024. Ausspioniert mit Standortdaten. https://interaktiv.br.de/ausspioniert-mit-standortdaten/en/.

[110] Erik Rye and Dave Levin. 2024. Surveilling the Masses with Wi-Fi-Based Positioning Systems. In *IEEE Symposium on Security and Privacy (SP)*.

[111] Piotr Sapiezynski, Radu Gatej, Alan Mislove, and Sune Lehmann. 2015. Opportunities and challenges in crowdsourced wardriving. In *Proceedings of the Internet Measurement Conference (IMC)*.

[112] Narmeen Shafqat, Nicole Gerzon, Maggie Van Nortwick, Victor Sun, Alan Mislove, and Aanjhan Ranganathan. 2023. Track You: A Deep Dive into Safety Alerts for Apple AirTags. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*.

[113] Singlespot. 2024. Singlespot - Pricing, Reviews, Data & APIs. https://datarade.ai/data-providers/singlespot/profile.

[114] Skyhook. 2024. Skyhook Wi-Fi Location. https://www.skyhook.com/wifi-location-solutions.

[115] Sooel Son, Daehyeok Kim, and Vitaly Shmatikov. 2016. What Mobile Ads Know About Mobile Users.. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[116] Statista. 2024. Mobile Android Version Market Share Worldwide 2018-2024. https://www.statista.com/statistics/921152/mobile-android-version-share-worldwide/.

[117] Ryan Stevens, Clint Gibler, Jon Crussell, Jeremy Erickson, and Hao Chen. 2012. Investigating user privacy in android ad libraries. In *Workshop on Mobile Security Technologies (MoST)*.

[118] Swrve. 2024. Swrve Geo SDK Integration Guide. https://docs.swrve.com/developer-documentation/integration/swrve-geo-sdk/.

[119] Byron Tau. 2023. FBI Once Bought Mobile-Phone Data for Warrantless Tracking. Other Agencies Still Do. https://www.wsj.com/articles/fbi-once-bought-mobile-phone-data-for-warrantless-tracking-other-agencies-still-do-ad65ebe9.

[120] Marcos Tileria and Jorge Blasco. 2022. Watch over your tv: a security and privacy analysis of the android tv ecosystem. In *Proceedings on Privacy Enhancing Technologies*.

[121] Narseo Vallina-Rodriguez, Jon Crowcroft, Alessandro Finamore, Yan Grunenberger, and Konstantina Papagiannaki. 2013. When assistance becomes dependence: characterizing the costs and inefficiencies of A-GPS. In *ACM SIGMOBILE Mobile Computing and Communications Review*.

[122] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S Cardoso, and Frank Piessens. 2016. Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms. In *Proceedings of the Asia conference on computer and communications security (ASIA CCS)*.

[123] Haoyu Wang, Zhe Liu, Jingyue Liang, Narseo Vallina-Rodriguez, Yao Guo, Li Li, Juan Tapiador, Jingcun Cao, and Guoai Xu. 2018. Beyond Google Play: A Large-Scale Comparative Study of Chinese Android App Markets. In *Proceedings of the Internet Measurement Conference (IMC)*.

[124] Jice Wang, Yue Xiao, Xueqiang Wang, Yuhong Nan, Luyi Xing, Xiaojing Liao, JinWei Dong, Nicolas Serrano, Haoran Lu, XiaoFeng Wang, et al. 2021. Understanding malicious cross-library data harvesting on android. In *Proceedings of the USENIX Security Symposium*.

[125] Fengguo Wei, Sankardas Roy, Xinming Ou, and Robby. 2018. Amandroid: A precise and general inter-component data flow analysis framework for security vetting of android apps. In *ACM Transactions on Privacy and Security (TOPS)*.

[126] Mira Weller, Jiska Classen, Fabian Ullrich, Denis Waßmann, and Erik Tews. 2020. Lost and found: stopping bluetooth finders from leaking private information. In *Proceedings on Security and Privacy in Wireless and Mobile Networks (WiSec)*.

[127] WiGLE. 2024. WiGLE All the Networks. Found by Everyone. https://wigle.net.

[128] WIRED. 2024. Anyone Can Buy Data Tracking US Soldiers and Spies to Nuclear Vaults and Brothels in Germany. https://www.wired.com/story/phone-data-us-soldiers-spies-nuclear-germany/.

[129] Wired. 2024. Jeffrey Epstein Island Visitors Data Broker Leak. https://www.wired.com/story/jeffrey-epstein-island-visitors-data-broker-leak/.

[130] Arol Wright. 2021. Android 12 no longer needs your location to scan nearby Bluetooth devices. https://www.xda-developers.com/android-12-location-scan-nearby-bluetooth-devices/.

[131] Senator Ron Wyden. 2024. Wyden Reveals Phone Data Used to Target Abortion Misinformation at Visitors to Hundreds of Reproductive Health Clinics. https://www.wyden.senate.gov/news/press-releases/wyden-reveals-phone-data-used-to-target-abortion-misinformation-at-visitors-to-hundreds-of-reproductive-health-clinics.

[132] Yue Xiao, Chaoqi Zhang, Yue Qin, Fares Fahad S Alharbi, Luyi Xing, and Xiaojing Liao. 2024. Measuring Compliance Implications of Third-party Libraries Privacy Label Disclosure Guidelines. In *Conference on Computer and Communications Security (CCS)*.

[133] Tingfeng Yu, James Henderson, Alwen Tiu, and Thomas Haines. 2024. Security and Privacy Analysis of Samsung's {Crowd-Sourced} Bluetooth Location Tracking System. In *Proceedings of the USENIX Security Symposium*.

[134] Jiexin Zhang, Alastair R. Beresford, and Stephan A. Kollmann. 2019. LibID: reliable identification of obfuscated third-party Android libraries. In *Proceedings of the SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*.

[135] Fanghua Zhao, Linan Gao, Yang Zhang, Zeyu Wang, Bo Wang, and Shanqing Guo. 2018. You are where you app: An assessment on location privacy of social applications. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*.

[136] Kaifa Zhao, Xian Zhan, Le Yu, Shiyao Zhou, Hao Zhou, Xiapu Luo, Haoyu Wang, and Yepang Liu. 2023. Demystifying Privacy Policy of Third-Party Libraries in Mobile Apps. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*.

[137] Qingchuan Zhao, Chaoshun Zuo, Giancarlo Pellegrino, and Li Zhiqiang. 2019. Geo-locating Drivers: A Study of Sensitive Data Leakage in Ride-Hailing Services. Proceedings of the Network and Distributed System Security Symposium (NDSS).

[138] Shuang Zhao, Xiapu Luo, Bo Bai, Xiaobo Ma, Wei Zou, Xinliang Qiu, and Man Ho Au. 2016. I know where you all are! exploiting mobile social apps for large-scale location privacy probing. In *Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part I 21*.

Appendix

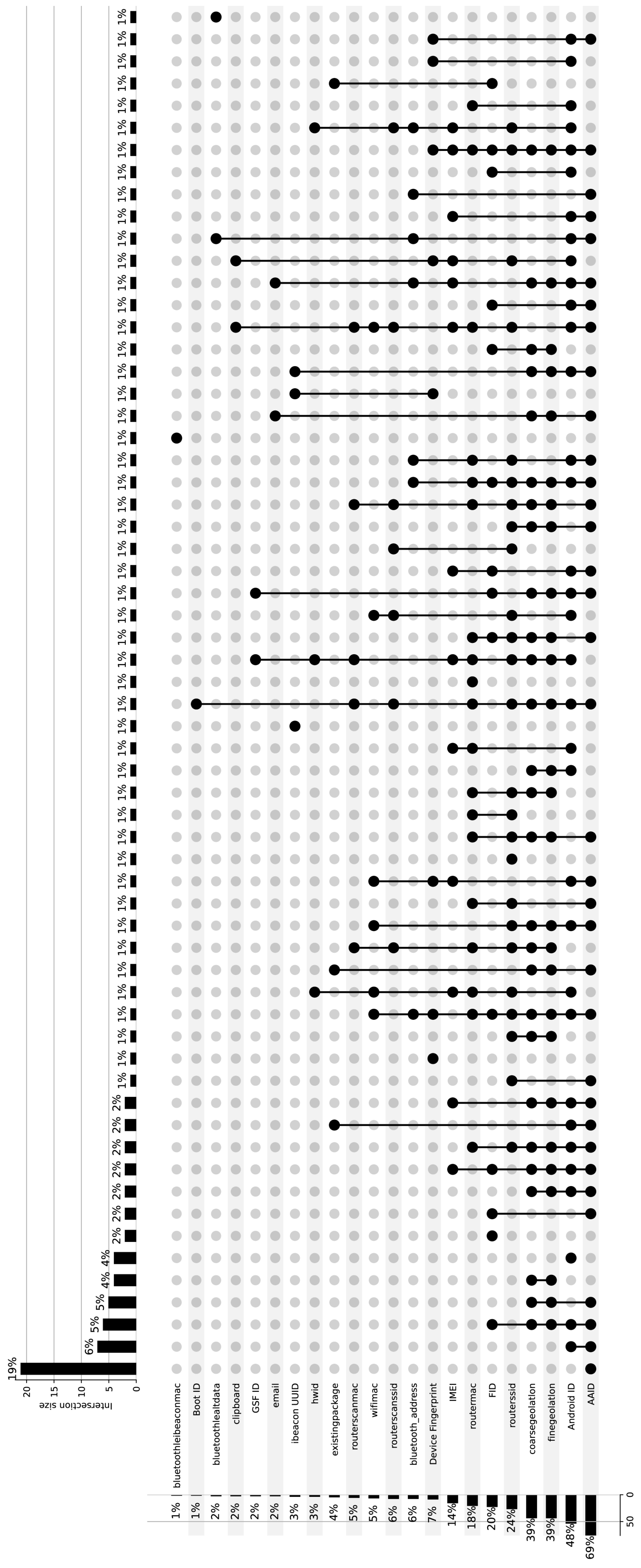

Figure 6: Upset plot of all the identifiers collected by the SDKs.

Table 6: All 52 SDKs in our dataset and their purposes.

SDK Name	# Apps	Purpose Type			
		Analytics	Location	Advertising	Profiling
AltBeacon	4,022	✓	✓		
Adobe Experience Platform	1,328	✓			
Kochava	1,117	✓	✓	✓	
Salesforce Marketing Cloud	1,080	✓	✓	✓	
Estimote	510	✓	✓		
LeanPlum	456	✓	✓		✓
Gimbal	396	✓	✓	✓	
Radius Networks	369	✓		✓	
mParticle	367	✓			
Ad4Screen	198			✓	
Kontakt	194	✓			
CueAudio	190	✓	✓		
Swrve	153	✓	✓		✓
Reveal Mobile	109	✓	✓		
Exponea	99	✓			
Radar	93	✓	✓		
IndoorAtlas	92	✓	✓		
SignalFrame	89		✓		
Bazaarvoice	88	✓	✓		
Huq Sourcekit	81	✓	✓		
Yinzcam Sobek	80	✓	✓	✓	
BlueKai (acquired by Oracle)	73	✓			
Cuebiq	73	✓	✓		
Rover SDK	50	✓	✓	✓	
Coulus Coelib	47			✓	
Colocator	40	✓	✓		
X-Mode	38	✓	✓		
Zendrive	34	✓	✓		✓
Dynamic Yield	27	✓			
Pilgrim by Foursquare	25		✓		
Sense360	24	✓	✓		
Locuslabs	23	✓	✓		
InMarket	23	✓	✓		
Singlespot	18	✓			
Roximity	17		✓		
Zapr	17	✓	✓	✓	
Swirl	16	✓	✓		
Bluecats	14		✓		
Areametrics	8	✓			
OpenLocate	8		✓		
Point Inside	7	✓	✓		
PredicIO	6	✓	✓		✓
MOCA	5	✓	✓		✓
BeaconsInSpace (Fysical)	5		✓		
Unacast Pure	5		✓		
Woosmap SDK	5	✓	✓		
Sensoro	4	✓	✓		
Signal360	4	✓			
Placer	4	✓	✓		✓
Proximi.io	3	✓	✓		
Tamoco	3	✓	✓		
pulseid	1	✓	✓		

Table 7: Description of PII types considered in this study. Abbreviations used: Pers - Persistent; Reset - Resettable.

Identifier	Pers	Reset	Definitions
IMEI	x		Unique hardware ID for the mobile device. Available only on Android 9 or below
WiFi MAC	x		Unique hardware ID for the WiFi interface.
HWID	x		Unique hardware ID assigned by the manufacturer.
GSF ID	x		Unique ID tied to the Google account on the device.
Boot ID	x		Unique identifier representing the device's boot session; resets only after a factory reset.
Email	x		User's email address.
Device Fingerprint			Non-unique Collection of attributes (e.g., OS version, browser) identifying the device..
Android ID	x		App-specific ID that resets with a factory reset.
AAID		x	Advertising ID provided by Google for specialized for advertising. Can be reset through device settings.
Firebase ID		x	App-specific ID tied to a specific app instance. It resets when the app is uninstalled or its data is cleared.
Bluetooth Device Name		x	User-defined, resettable, and non-unique name for Bluetooth discovery.
Bluetooth iBeacon MAC	x		Unique hardware address for BLE proximity detection.
iBeacon UUID	x		Unique ID for iBeacon devices in proximity-based services.
Router MAC	x		Unique mac address for the connected WiFi router.
Router Scan MAC	x		Persistent hardware address for nearby routers detected in WiFi scans,
Router Scan SSID	x		Names of nearby routers detected in WiFi scans,
Router SSID	x		Name of the currently connected router.
Coarse Geolocation	x		Co-ordinates of approximate device location derived from network data.
Fine Geolocation	x		Co-ordinates of precise GPS-based location of the device.

Table 8: Example of Beacon SDK code and network signatures.

SDK Name	Package Names	Domains / Endpoints
AltBeacon	org.altbeacon.beacon., com.altbeacon.beacon., org.altbeacon.bluetooth.	data.altbeacon.org
Radius Networks	com.radiusnetworks.	proximitykit.radiusnetworks.com
Estimote	com.estimote.	*.estimote.com
Gimbal	com.gimbal.android.	analytics*.gimbal.com, api.gimbal.com, sdk-info.gimbal.com
Kontakt	com.kontakt.sdk.android.	kontakt.io
Reveal Mobile	com.stepleaderdigital.reveal.	*.revealmobile.com rvl.wral.com
SignalFrame	com.wirelessregistry.observersdk.	*.wirelessregistry.com
IndoorAtlas	com.indooratlas.android.sdk.	ipsws.indooratlas.com
Rover SDK	io.rover.	*.rover.io

Table 9: Beacon SDK search keywords and attribution signals.

Technology	Search Keywords	Attribution Signals
BLE	"beacon SDK", "proximity SDK", "BLE SDK", "Bluetooth tracking SDK", "BLE advertising SDK", "Bluetooth beacon SDK", "RSSI scanning SDK"	Presence of BLE scanning APIs, class names associated with BLE scanning, declared BLE-related permissions in AndroidManifest.xml, references to hardcoded UUIDs for beacon identification, extraction of RSSI values for signal strength analysis
WiFi	"WiFi beacon SDK", "proximity SDK", "WiFi tracking SDK", "WiFi positioning SDK", "WiFi scanning SDK"	Presence of WiFi scanning APIs, class names related to WiFi scanning, declared WiFi-related permissions in AndroidManifest.xml, analysis of hardcoded SSIDs, references to known WiFi telemetry endpoints
Location	"geofencing SDK", "proximity SDK", "location tracking SDK", "indoor positioning SDK", "RTLS SDK", "asset tracking SDK"	Presence of location-related APIs, class names associated with location services, declared location-related permissions in AndroidManifest.xml, embedded geofencing configuration files, hardcoded location-based UUIDs

Table 10: Android framework APIs (Java) that enable wireless beacon scanning.

Technology	Permissions	APIs
BLE	BLUETOOTH_SCAN BLUETOOTH_CONNECT ACCESS_FINE_LOCATION	android.bluetooth.BluetoothAdapter/startLeScan android.bluetooth.BluetoothLeScanner/startScan android.bluetooth.BluetoothDevice/connectGatt android.bluetooth.BluetoothGatt/discoverServices android.bluetooth.BluetoothLeAdvertiser/startAdvertising
WiFi	CHANGE_WIFI_STATE ACCESS_WIFI_STATE ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION	android.net.wifi.WifiManager/getScanResults android.net.wifi.WifiManager/startScan android.net.wifi.WifiManager/getConnectionInfo android.net.wifi.WifiManager/setWifiEnabled android.net.wifi.WifiManager/reconnect
Location	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION FOREGROUND_SERVICE	android.location.LocationManager/getLastKnownLocation android.location.LocationManager/requestLocationUpdates android.telephony.TelephonyManager/getAllCellInfo android.telephony.TelephonyManager/getCellLocation android.location.LocationManager/addProximityAlert

Table 11: Market share of the top-15 SDKs per top-10 app category.

SDK Name	# Apps	Lifestyle	Shopping	Sports	Business	Travel & Local	Finance	News & Magazines	Education	Entertainment	Weather
AltBeacon	4024	18% (721)	6% (240)	3% (137)	12% (498)	8% (338)	3% (133)	1% (59)	8% (302)	3% (114)	1% (32)
Adobe Experience Platform	1328	3% (45)	7% (89)	9% (120)	4% (47)	5% (65)	13% (179)	16% (212)	1% (13)	7% (97)	18% (245)
Kochava	1118	3% (34)	3% (36)	7% (73)	2% (20)	2% (18)	5% (60)	2% (25)	4% (42)	7% (75)	0% (2)
Salesforce Marketing Cloud	1080	9% (98)	23% (244)	3% (33)	10% (103)	8% (83)	18% (197)	2% (24)	1% (12)	2% (24)	0% (0)
Estimote	510	34% (174)	3% (14)	1% (6)	12% (60)	13% (65)	1% (6)	0% (2)	13% (64)	3% (14)	0% (0)
LeanPlum	456	3% (15)	7% (33)	1% (6)	2% (10)	14% (63)	5% (21)	1% (3)	14% (66)	2% (8)	0% (2)
Gimbal	396	10% (40)	6% (23)	42% (167)	1% (5)	1% (5)	20% (80)	3% (10)	5% (21)	6% (24)	0% (0)
Radius Networks	369	44% (162)	6% (23)	2% (9)	6% (21)	4% (13)	1% (3)	1% (3)	3% (12)	2% (7)	0% (1)
mParticle	367	3% (11)	6% (21)	6% (21)	3% (11)	16% (59)	8% (30)	14% (53)	1% (4)	9% (32)	0% (0)
Ad4Screen	198	5% (10)	19% (38)	3% (5)	2% (3)	7% (14)	12% (23)	29% (58)	1% (2)	2% (3)	0% (0)
Kontakt	195	9% (17)	15% (30)	7% (13)	21% (40)	7% (13)	2% (4)	2% (3)	4% (7)	5% (9)	1% (1)
CueAudio	190	0% (0)	0% (0)	98% (187)	0% (0)	0% (0)	0% (0)	0% (0)	0% (0)	1% (2)	0% (0)
Swrve	153	2% (3)	8% (13)	3% (4)	0% (0)	7% (11)	5% (8)	1% (2)	2% (3)	5% (7)	1% (1)
Reveal Mobile	109	0% (0)	0% (0)	0% (0)	0% (0)	0% (0)	0% (0)	13% (14)	0% (0)	1% (1)	86% (94)
Exponea	99	4% (4)	38% (38)	0% (0)	2% (2)	10% (10)	15% (15)	0% (0)	3% (3)	4% (4)	0% (0)
Total apps per category		12% (1385)	9% (999)	8% (935)	8% (903)	7% (847)	7% (795)	5% (600)	5% (562)	4% (473)	4% (420)

Table 12: Data collected by Vizbee.

Package Name	Installs	AAID	Router MAC	Router SSID	Android ID	Coarse Geolocation
com.gotv.nflgamecenter.us.lite	120.1M	x	x	x	x	x
com.cnn.mobile.android.phone	50M	x	x	x	x	
com.handmark.sportcaster	18.6M	x	x	x	x	
com.turner.tnt.android.networkapp	7.6M	x	x	x	x	
com.nfl.fantasy.core.android	7.4M	x	x	x	x	x
com.neulion.android.tablet.nfl.wnfln	3.9M	x	x	x	x	x
com.turner.trutv	1.7M	x	x	x	x	
com.fox.weather	959.7K	x	x	x	x	x
com.gannett.local.library.news.kare	198.8K	x	x	x	x	x
com.raycom.wtol	194.1K	x	x	x	x	x
com.gannett.local.library.news.wxia	151.0K	x	x	x	x	x
com.gannett.local.library.news.kxtv	54.7K	x	x	x	x	x

Table 13: SDKs that present rationale behind dangerous permission requested using shouldShowRequestPermissionRationale().

SDK	% Ctx	C. Loc	F. Loc	B. Loc
Estimote	75%	✓		
InMarket	52%			✓
Radar	42%	✓	✓	
Singlespot	55%		✓	
Salesforce Marketing Cloud	54%		✓	

Abbreviations Used: SDK - Software Development Kit, % Ctx - % of apps that Show Context, C. Loc - Access Coarse Location, F. Loc - Access Fine Location, B. Loc - Access Background Location.

```
POST /track/json HTTP/1.1^M
User-Agent: Dalvik/2.1.0 (Linux; U; Android 12;
    AOSP on sargo Build/SP2A.220505.008)^M
Content-Type: application/json; charset=UTF-8^M
Host: control.kochava.com^M
Content-Length: 1729^M
^M

"action":"install","kochava_app_id":"kobiubiuclub
    -3b00","kochava_device_id":"
    KA31101723813375tdacae51f18644fed9ef9b8467b07d2be
    ","sdk_protocol":"14","sdk_version":"
    AndroidTracker 3.11.0","nt_id":"bc770-1-
    acccf2f1-33da-4dbc-80bf-6a5981859f3f","data
    ":{"screen_brightness":0.2902,"
    device_orientation":"portrait","volume":0.32,"
    aaid": "X7dD24S2-CcD4-4Cfv-hhgg-bbbb7yk245f6"
    ,"device":"AOSP on sargo-Android","disp_h
    ":2220,"disp_w":1080,"package":"com.biubiuclub
    .biubiuclubchat","installed_date":1723813319,"
    app_name":"BiubiuClub","app_version":"291","
    app_short_string":"2.9.1","os_version":"
    Android

"network_conn_type":"wifi","ssid":, "routerssid:
    ABC-WIFI", "bbsid": "routermac : XX:XX:XX:XX:
    XX:XX", "network_metered":false,"nvp":0,"
    carrier_name":"","usertime":1723813375,"uptime
    ":0.945,"state_active":true,"
    app_limit_tracking":false,"platform":"android
    "},"sdk_id":"c946-s88450-","send_date
    ":"2024-08-16T13:02:56.954Z"
```

Figure 7: The Kochava SDK in com.biubiuclub.biubiuclubchat collects AAID and router MAC address and transmits them to control.kochava.com

```
POST /events/v3 HTTP/1.1
x-dynatrace: MT_3_4_2021016446_1-0_36d6e79e-
"records": [
    {
      "app_session_id": "474c6944-823d-4da3-b154
          -001b272a6f59",
      "event_type": "bulk_localization_event",
      "localizations": [
        {
          "ad_tracking_enabled": "true",
          "Ad_id": "ccccccc-cccc-cccc-cccc-
              ba1bccccc0a",
          "app_package_name": "br.com.bancobmg.
              bancodigital.atletico",
          "app_state": "FOREGROUND",
          "fingerprint": {
            "elapsed_ts": "1589677460",
            "environment_scan": {
              "environment_state": "outdoor",
              "timestamp": "1723858094723"
            },

"gps_fix": {
                "altitude": "721.5999755859375",
                "bearing": "227.02573",
                "bearing_acc": "45",
                "gps_acc": "12.75",
                "lat": "30.3368394",
                "lng": "-2.7705837",
                "provider": "fused",
                "speed": "2.1049643",
                "speed_acc": "1.5",
                "vertical_acc": "1.5948421"
              },
              "gps_ts": "1723858073227",
              "mock_location": "false",
              "precise_location": "true"
            },
            "wifi_scan": {
              "ap_measures": [
                {
                  "ap_ts": "1723858075286",
                  "auth": "false",
                  "bssid": "XX:XX:XX:XX:XX:XX",
                  "channel_width": "20_mhz",
                  "con": "false",
                  "frequency": "2462",
                  "level": "-60",
                  "ssid": "ABC-WIFII",
                  "venue_name": "",
                  "wifi_rtt_responder": "false"
                },
```

Figure 8: The Incognia SDK in br.com.bancobmg.bancodigital.atletico collects AAID, GPS coordinates, and WiFi scan data and transmits them to service2.us.incognia.com.