AppCensus

149 New Montgomery Street
4th Floor
San Francisco, CA 94105

1,000 Mobile Apps in Australia:

A Report for the ACCC

September 24, 2020

www.appcensus.io

Executive Summary

This report was prepared for the Australian Competition and Consumer Commission (ACCC) and contains findings of an analysis conducted from June–July 2020 of 1,000 of the most popular mobile Android apps ("apps") in Australia.[1]

The purpose of this analysis was to examine both the extent to which these apps collected user information[2] during the testing period, which apps transmitted a user's information to third parties, and if so, who these third parties were. Android apps were the focus of this analysis because the Android operating system is open source: our testing relies on us being able to modify the operating system so that when third-party mobile apps are run, we can detect whenever they attempt to access sensitive user information and whether they attempt to transmit it over the Internet (regardless of their use of encryption or other obfuscation techniques). Unlike other app analysis techniques, by instrumenting the operating system itself, our monitoring is generally not detectable by the apps that we are testing. Simply put: we did not examine apps on Apple's iOS because we did not have access to its source code, and therefore a comparable level of analysis is not possible.[3]

The apps subject to our analysis were divided by the ACCC among three categories: 103 *Health* apps, 100 *Kids* apps, and 797 *Other* apps. We tested these apps by running them on Android mobile phones located in Australia and then observed whether they accessed certain user information stored on the devices (described in Section 3), and if so, if and to whom they transmitted it. In this manner, we documented the transmissions of user information that we observed during our testing. However, because many of the behaviors of interest are non-deterministic in nature—running the same app with the same input may not result in transmissions to the same third parties—we also estimated apps' *potential* to transmit user information to various third parties based on the prevalence of third-party software components that may transmit information to their parent companies' servers. By testing these apps in this manner, we document the following in this report:

[1] Based on ranking and active users, the top 1,000 most popular Android apps consist of top apps on the Google Play Store across all categories and at least 100 top apps in both the *Fitness* and *Health* categories (`Health apps') and in the *Education*, *Games* and *Animation and Comics* categories that are targeted to children aged 13 and under (`Kids apps').

[2] For a description of what was defined as "user information," see Section 3.

[3] We note that worldwide, Android market share leads iOS by roughly three-to-one (as of this writing): https://web.archive.org/web/20200817231546/https://gs.statcounter.com/os-market-share/mobile/worldwide.

1. **The types of unique user information that each app accessed and transmitted, as well as analysis of apps' capabilities and data-sharing practices (Section 4.1):**

   During our testing, the most prevalent type of user information transmitted by apps to third parties was the user's Android Advertising ID (AAID):[4] over 60% of Health and Other apps and over 40% of Kids apps transmitted it (Figure 1).

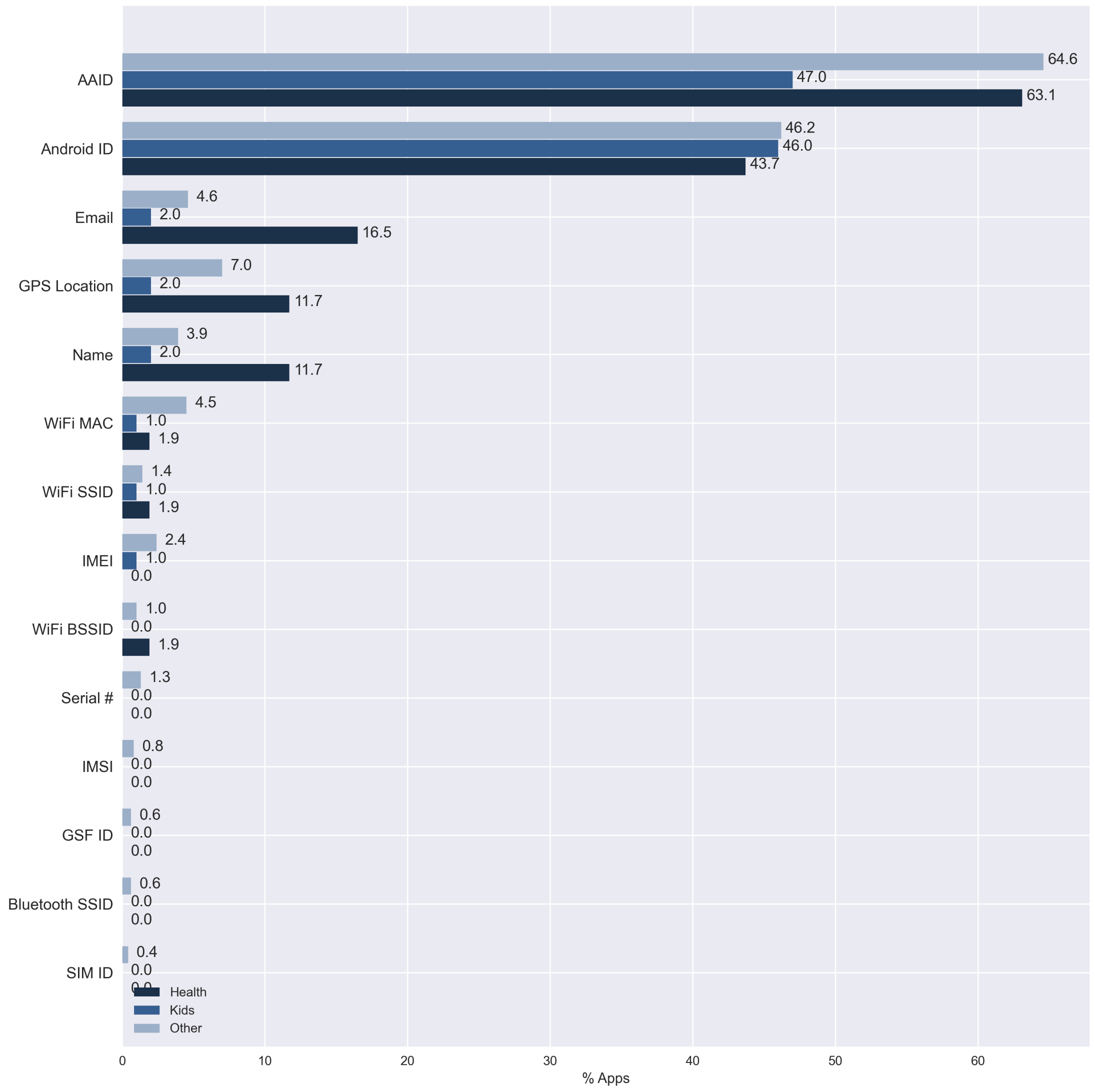


Figure 1: Percentage of apps in each category that transmitted various data types.

[4]The Android Advertising ID (sometimes known as the "Google Advertising ID" or GAID) is a resettable identifier, meaning that users can go through the system settings to reset it to a new value. Google's developer guidelines require that this is the only identifier that is used for advertising and analytics purposes.

The AAID is an example of a persistent identifier, which is a unique number that allows an individual's behaviors to be tracked over time: in much the same way that an individual's routine and shopping habits could be inferred if their vehicle's license plate was recorded in front of all the businesses where it was observed, persistent identifiers allow recipients to track an individual's online activities, including what apps they use, and what they do within those apps. In the case of the AAID (sometimes referred to as the "Google advertising identifier" or GAID), Google's documentation specifies that the AAID "must only be used for advertising and user analytics."[5] Given that the AAID was the most prevalent type of user information that was shared with third parties, it is clear that much of this data sharing is in the service of advertising and analytics.

The second most common type of user information that we observed being transmitted by apps during testing was the Android ID (sometimes known as the "SSAID").[6] The Android ID, a distinct identifier from the aforementioned AAID, can only be reset with a factory reset of the device. However, in Android version 8 and above, the Android ID is unique for each app *developer*: if multiple apps from different developers transmit the Android ID to the same third party—something that is very common—the recipient third party will receive different Android IDs for each app, and therefore will not be able to (easily) correlate an individual user's behavior across apps. This identifier was collected by over 40% of all apps, regardless of category.

Finally, we observed that location data, in the form of precise GPS coordinates (i.e., accurate to 100m), was transmitted by almost 12% of Health apps, 7% of Other apps, and 2% of Kids apps during testing. However, we should note that every device connected to the Internet automatically transmits its IP address, which can be used to infer coarse-grained location information (e.g., city-level).

Just because we did not observe an app transmit user information to a data recipient during testing, does not mean that it might not occur under different conditions. Thus, to attain another measure of the likely recipients of user information from mobile apps, we examined the software development kits (SDKs) present in the apps (Figure 2).

SDKs are third-party software components that app developers bundle within their apps to provide certain functionality. While some of this functionality may be in the service of providing primary app features, other SDKs may collect user information for secondary purposes. Because third-party SDKs embedded in an app have access to the same data and system resources as the host app, they can potentially collect a lot of sensitive user data; in some cases, the app developer may not even be aware of the data being collected by a third-party SDK. Therefore, measuring the prevalence of the most popular SDKs among Android apps provides a metric for *potential* data collection from mobile apps.

As can be seen in Figure 2, Google's SDKs were identified in 92% of the mobile apps that we analyzed. This was followed by Facebook, which was identified as having SDKs in 61% of the apps analyzed. We observed that most of many popular SDKs that we identified are designed to collect user information for advertising purposes.

[5] https://web.archive.org/web/20200528143805/https://play.google.com/about/monetization-ads/ads/

[6] https://web.archive.org/web/20200702180532/https://android-developers.googleblog.com/2017/04/changes-to-device-identifiers-in.html

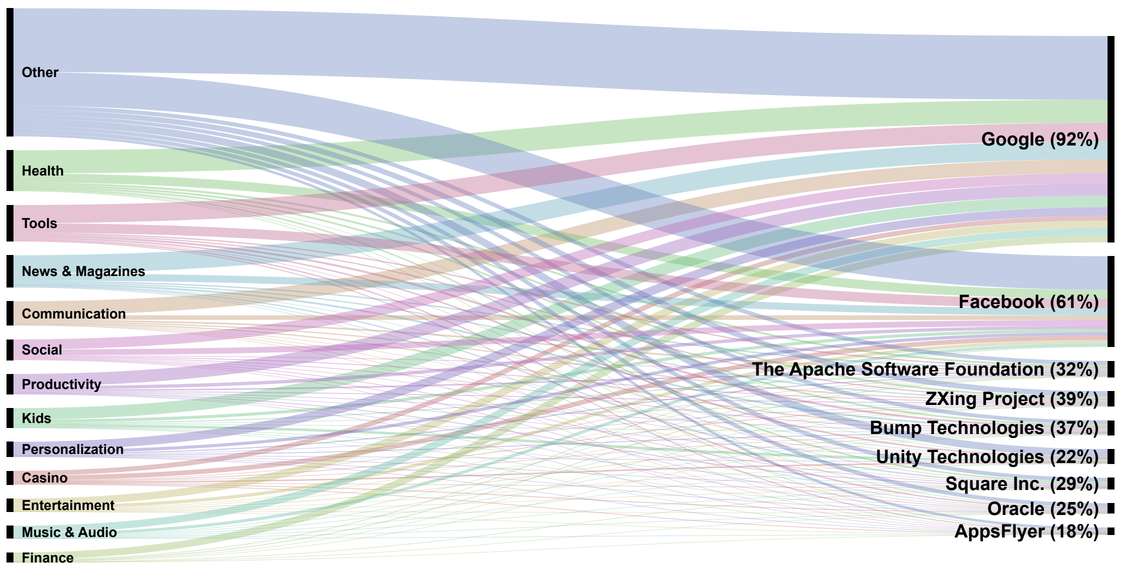


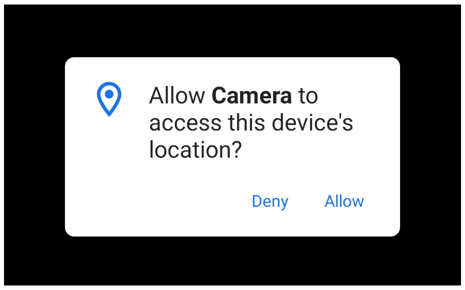
Figure 2: App categories containing SDKs from various companies.

Because many of these SDKs automatically collect user information as soon as the host app is started, this may run afoul of various privacy regulations that require a legal basis for data collection (e.g., consent) or allow users to opt out of data collection. To comply with these requirements, many SDKs offer app developers options to disable automatic data collection (e.g., so that users can first consent): for the few that we examined, we observe relatively low usage rates among the apps that we tested. Thus, it is likely that most data collection occurs by default, and that in many cases, the collected information will be used for secondary purposes.

2. **The methods by which user information is accessed by each app (Section 4.4)**

   The Android operating system employs a "permissions system" so that when an app attempts to access a protected resource for the first time, the user will be given the opportunity to allow or deny that request (Figure 3).

   Not only does the permissions system require user consent before an app can access certain device resources, but it also is supposed to prevent apps from accessing certain types of user information altogether, based on the potential for privacy abuses. Nonetheless, we observed dozens of apps transmitting device WiFi MAC addresses, which are hardware-based identifiers that can be used for long-term tracking, and cannot easily be reset by the user (e.g., in the way that cookies stored in a web browser can). Google prohibited Android app developers from collecting the WiFi MAC address starting in 2015, and even included technical means to enforce this policy. Yet, we specifically observed that Umeng's SDK, a subsidiary of Alibaba, exploits a security vulnerability in Android that allows any app containing their SDK to collect this identifier, despite Google's policies. (Umeng's SDK was observed doing


Allow **Camera** to access this device's location?

Deny Allow

Figure 3: Prompt generated by the Android permissions system.

this in over 30 apps that we tested.) We reported this vulnerability to Google, but as of this writing, this issue still persists. As a result, consumers remain vulnerable to this type of data collection, which may result in consumer confusion over apps' privacy behaviors, or worse, may render some of Android's privacy controls effectively useless.[7]

Google provides privacy controls for Android users that allow them to reset the value of the AAID, in much the same way that web browsers allow users to delete cookies: by deleting these identifiers, future online activities cannot easily be matched with the data that was previously collected from that user. However, the ability to reset the AAID is only effective so long as it is the only identifier collected: if another identifier is collected alongside the AAID, and the user elects to reset the AAID (e.g., to prevent profiling), then the recipient can use the identifier to not only correlate the user's future activities with previously-collected data, but also to determine that this particular user reset the AAID. We examined the extent to which the AAID was collected alongside other identifiers (thereby undermining the system-wide privacy control), which would allow a data recipient to continue tracking the same user, even after that user resets their AAID. We observed that 32% of all apps transmitted other identifiers alongside the AAID, which was split roughly evenly across all app categories.

3. **All permissions requested by apps within the categories provided (Appendices B.2, C.2, D, and F).**

   We analyzed each app to detect which permissions they *could* request when run, and then during testing we examined which of those permissions they *actually* used. In many cases, app developers request more permissions than their apps actually need, which puts user data at risk of being accessed by third-party SDKs unnecessarily. We show several examples of apps requesting permissions that may defy users' expectations, as well as listing the apps that requested the most permissions.

   The most commonly requested permissions governed the ability to read and write data to the device's shared storage: 55% and 67% of Health apps, respectively; 41% and 67% of Kids apps, respectively; and 64% and 78% of Other apps, respectively. This permission enables an app to read and write data that has been stored by other

[7] `https://web.archive.org/web/20200902193655/https://blog.appcensus.io/2019/02/14/ad-ids-behaving-badly/`

apps, which includes the device's photo library. While there are many legitimate uses for this ability, we have previously documented instances where apps abused this ability to collect user data that they were not permitted to collect [10].

The second most commonly requested permissions governed access to location data, both fine-grained and coarse-grained.[8] For example, during testing, we observed that 54% of Health apps, 5% of Kids apps, and 40% of Other apps accessed fine-grain location data. We hypothesize that the relatively low rate of access to this data amongst Kids apps may be due to privacy regulations in other jurisdictions (i.e., apps developed for international audiences may elect to comply with the most stringent regulations). At the same time, we noted that of the apps accessing fine-grain GPS location data during the testing period, many fewer were observed transmitting it. We hypothesize that this discrepancy may be due to the fact that we only flagged transmissions of location coordinates if they were accurate to within 100m of the device's true location. Thus, it is possible that some apps may be accessing the device's GPS sensor and then transmitting location data with a lower degree of precision.

4. **All third-party recipients of user information within the categories provided (Section 4.2), as well as for each app (Appendix F).**

   We examined the transmissions containing user information that we were able to identify, and in so doing, ranked the top recipients of information from the apps tested. The top ten third-party parent companies of data recipients (Figure 4) were Facebook (observed receiving user information from 40% of apps tested), AppsFlyer (15%), Unity (12%), Google (11%), Adjust (11%), Twitter (8%), Verizon (8%), Branch (7%), Amazon (4%), and Liftoff (4%). For each category that we analyzed, we provide examples of the 1) top 5 apps that transmitted the most types of device and user information and 2) the top 5 types of device and user information transmitted. Facebook alone received user information from a majority of all apps, including 13% of Kids apps, 40% of Health apps, and 44% of Other apps.

   Figure 4 depicts only the transmissions containing user information that we were able to identify, whereas Figure 5 depicts the top companies that we observed *communicating* with apps during testing (regardless of whether or not we were able to automatically detect the transmission of user information). We can see that Google's servers were contacted by 68% of the apps that we tested. In many cases, the transmissions indicated that the data would be used for crash reporting and analytics, however, over 20% of apps contacted Google servers clearly involved in advertising. This, still, is an undercount of the information that Google receives about apps installed on users' Android devices: through the Google Play Services components of the Android operating system, we observed that Google received information about all apps that were installed on the phones during the testing period, as well as user information, in the form of persistent identifiers.

[8]The distinction between fine-grain and coarse-grain location data is the difference between accessing the device's GPS sensor to get precise coordinates versus inferring the device's location based on network information (i.e., nearby WiFi and cellular networks).

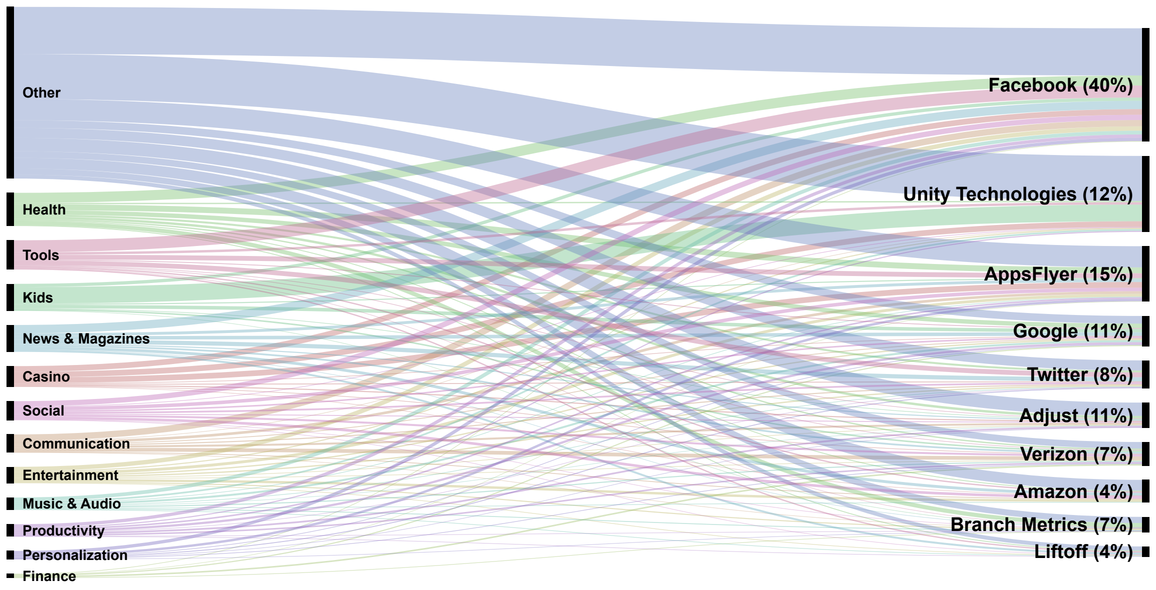


Figure 4: App categories *transmitting* user information to various companies during testing.

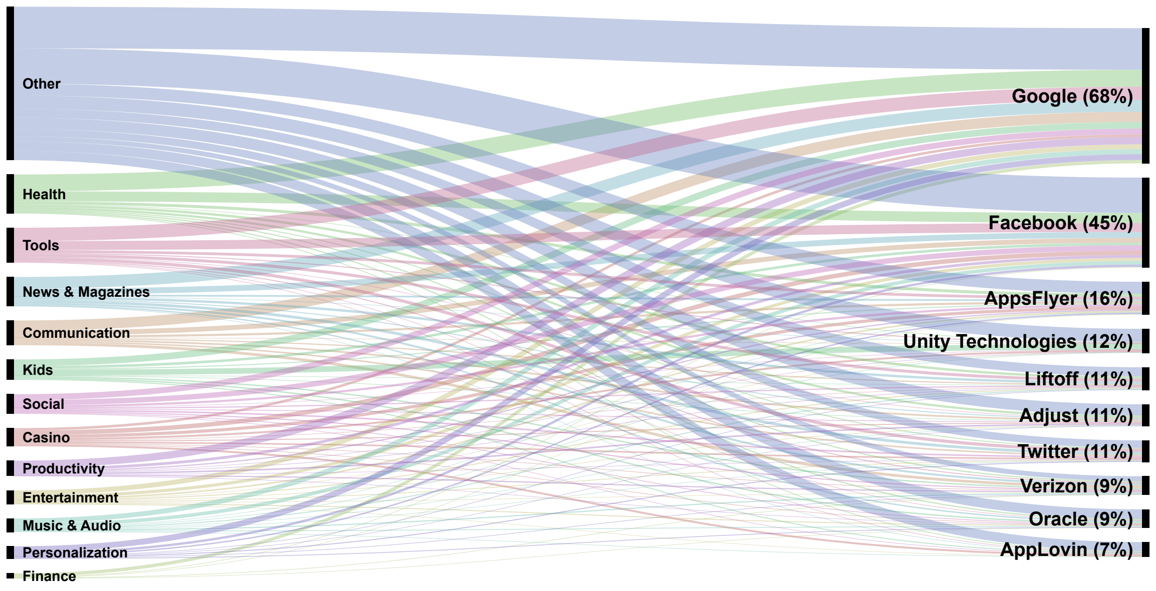


Figure 5: App categories *contacting* various companies during testing.

5. **We also present:**

    - A ranking of apps within each category based on the number of entities to which each app transmitted sensitive user information (Section 4.2 and Appendices B.1.2, C.1.2, and D).
    - Analysis of in-app privacy disclosures (Section 4.5).
    - Analysis of whether personal data was properly secured through the use of encryption (Section 4.6).
    - A ranking of the most prevalent recipients of user information (Section 4.2 and Appendices B.1.2, C.1.2, and D).

Disclaimer

This report was produced independently by AppCensus, without the sponsorship, endorsement or participation of any software provider discussed herein. At this time, AppCensus has not reported the results of its analysis, or sought input or comment, from any software provider discussed herein.

The results in this report are limited to the particular software versions and experimental conditions described. Please be aware that results could differ in other environments. Analysis of mobile phone apps is complex, and although we believe we have made reasonable efforts to provide an accurate report, we expressly disclaim all warranties, including without limitation the warranty of merchantability and fitness for a particular purpose, and we do not warrant that the contents of this report, or the findings or conclusions herein, are complete or error-free.

Table of Contents

List of Appendices

1 Introduction

In this report, we present the results of an analysis of 1000 Android apps (`apps') from June–July 2020, divided among three categories by the ACCC: 103 Health apps, 100 Kids apps, and 797 Other apps. During the testing period, we undertook three key areas of analysis:

1. We examined the apps' network traffic to detect the transmission of user information to third parties during the testing period (e.g., known persistent identifiers, contact information, location data, etc.).
2. We identified the third-party SDKs that were embedded within the apps, which have the potential to transmit user information to those third parties (and others).
3. We analyzed how the apps were making use of permission requests, which regulate access to sensitive user data (i.e., `user information,' which we define in Section 3). We specifically examined what permissions were requested by the apps and whether those permissions were actually used at runtime.

We performed our analysis on instrumented mobile phones located in Australia. Each app was installed on a test phone and executed while the operating system recorded whenever the app tried to perform an action that required a "dangerous" permission (which we define in Section 4.3). The permissions that an app may use are known *a priori*; during our testing, we determined which of these were actually used during app execution. Additionally, we monitored network traffic going in and out of the device, including traffic encrypted using TLS and "pinned" certificates. We searched this traffic for persistent identifiers, location information, and other sensitive data.

Background Information

What does this have to do with app users' data, user control, and privacy?

Mobile phones are personal devices that provide many ways for apps to learn information about their users. Mobile phones store and collect a wealth of sensitive user information, such as contacts, messages, photos, and location trails (i.e., records of the user's travels in the physical world). Additionally, a mobile app can collect information about how an individual uses it. This could include how frequently it is used, in addition to what actions a user takes in the app, what content is viewed—including which ads—and whether that content is successful at modifying user behavior (e.g., whether an ad worked). All of this data may be accompanied by one or more unique persistent identifiers, which attribute it to a specific individual. In many cases, apps access and collect user information and share it with third parties for purposes that are secondary to the functionality of the app. As a result, much of this data sharing occurs opaquely to consumers.

Our goal through this report is to document the following:

1. The types of unique user information that each app accessed and transmitted, as well as analysis of apps' capabilities and data-sharing practices (Section 4.1).
   (a) In Appendix F, we document the types of unique user information that each app transmitted and the SDKs of third parties that were identified as embedded within them, which allows us to identify other *potential* data recipients. Because of the way that Android is designed, every third-party SDK bundled within an app has access to the same user information that the app itself is able to access from the operating system. That is, if an app has been granted access to location data after the permissions system prompted the user, any third-party SDKs embedded within that app will also be granted access to location data. Thus, while our observations of transmissions during the testing period identify a set of data recipients at that point in time, the presence of third-party SDKs may indicate that other data recipients may receive user information if the app is run under different testing conditions.
   (b) In Section 4.1.2, we discuss the most prevalent third-party SDKs identified within the apps, the function of those SDKs, as well as the parent companies who could *potentially* receive user information from the SDKs.
   (c) In Section 4.1.3, we discuss apps' usage of their embedded SDKs' privacy-related configuration options and how they yield potential information about data-sharing between first- and third-party data recipients. This allowed us to draw conclusions about apps' treatment of user data.
2. The methods by which user information is accessed by each app (Section 4.4).
   (a) Whether any apps accessed user information by exploiting security vulnerabilities, thereby circumventing the Android permissions system (Section 4.4.1).
   (b) Whether any apps potentially violated Google Play Store developer policies by accessing or handling user information in prohibited ways (Section 4.4.2).
3. All permissions requested by apps within the categories provided (Appendices F, B.2, C.2, and D).
   (a) Whether we observed apps using these permissions during testing (Section 4.3.1 and Appendix F).
   (b) Whether use of those permissions seemed appropriate and consistent with app features (Section 4.3).
4. All third-party recipients of user information within the categories provided (Section 4.2), as well as for each app (Appendix F).
   (a) Full hostnames and IP addresses for all Internet servers observed receiving sensitive user information during testing (Appendix F).
   (b) Analysis of the most prevalent third-party recipients of user information from the apps, including hostnames, geolocations, and corporate information (Sections 4.1.2 and 4.2).

5. A ranking of apps within each category based on the number of entities to which each app transmitted sensitive user information (Section 4.2 and Appendices B.1.2, C.1.2, and D).
6. Analysis of in-app privacy disclosures (Section 4.5).
   (a) Whether data collection occurred prior to explicit user consent (Sections 4.1.3, 4.2.1, and 4.5).
7. Analysis of whether personal data was properly secured through the use of encryption (Section 4.6).
   (a) Examination of obfuscation techniques used to obscure the transmission of sensitive user information (Appendix A).
8. A ranking of the most prevalent recipients of user information (Section 4.2 and Appendices B.1.2, C.1.2, and D).

In Section 2, we describe our testing methodology. In the sections that follow, we describe the types of user information we examined and present our high-level analysis (Section 4), the limitations (Section 5), and bibliography (Section 6). In the appendices, we perform an additional analysis of selected obfuscated data transmissions observed during testing (Appendix A), followed by summary data for apps in each of the three categories (Appendices B, C, and D). The appendices conclude with a list of all apps (and versions) analyzed (Appendix E), as well as each app's testing results (Appendix F).

We caution that just because we observed that a permission was not used, or that a transmission of data did not occur, does not mean that it would not occur under different circumstances. Certain functionality may not have been triggered during the testing period. Moreover, apps may collect and transmit data using means that are not monitored by our instrumentation. In particular, any apps for which we observed no transmissions of sensitive data may be further scrutinized to determine if such techniques are in place (by inspecting the raw data associated with this report). Limitations of this research are noted in Section 5.

2 Methodology

Background Information

How do you examine the behaviors of mobile apps?

We examine an app from two perspectives. We look at what an app can *potentially* do and then what it *actually* does when we use it under realistic conditions. The former gives us an idea of the potential capabilities of the app. For example, we can learn what third parties it *may* try to contact based on the third-party SDKs contained within it. When we run the program and test it, we can observe what data is transmitted to what Internet servers and which permissions are actually used when accessing that data. We do this by running the app on a specially-instrumented mobile device, which precisely monitors what data is accessed and transmitted.

Why do you only look at Android apps?

Android is "open source," which means that the source code for it is freely available for others to use and modify.[a] Our instrumentation relies on us being able to modify the Android operating system so that we can monitor app execution without modifying the apps being tested. It also allows our instrumentation to go undetected by mobile apps in ways that other monitoring techniques might not. We cannot perform this level of monitoring on iOS simply because we do not have access to the iOS source code.

[a] https://developer.android.com/

Our approach to analyzing the behaviors of mobile apps relies on a combination of "static" and "dynamic" analysis. Static analysis refers to analyzing programs without running them, in order to detect the presence of specific instructions that *may* be executed when the program is run. For example, static analysis can be used to answer the question, "does the program include a particular function?" Static analysis is quick, because it does not involve interactively running the program. However, it is prone to false positives, as not all program code is reached during execution (i.e., "dead code"); code detected through static analysis may never get executed in practice. Dynamic analysis, on the other hand, refers to running programs to directly observe their behaviors. For example, dynamic analysis can answer the question, "what functions does the program actually use?" Dynamic analysis more realistically models program behavior, as the program is executed in a testing environment that is designed to model real-world usage. However, it is prone to false negatives: program code not executed during the testing period may be executed under different conditions. Dynamic analysis is also desirable because it does not yield false positives: conclusions are observations of *actual* program behavior during the testing period. We discuss the limitations of our methods in more detail in Section 5.[9]

We performed dynamic analysis on the apps to examine whether user information would be transmitted over the Internet during the course of realistic app usage. During testing, our automated system installed apps on smartphones and then human users interacted with them. We examined the network traffic generated by the apps in order to detect the transmission of user information (discussed in detail in Section 3 of this report). Additionally, we ran the apps with our own modified version of the operating system, which included additional instrumentation to monitor how apps attempted to access sensitive data stored on the device, including usage of the Android permissions system. Finally, we performed static analysis of the apps to identify bundled third-party software development kits (SDKs) and the use of various privacy-related functions and settings.

The test devices were all Google Pixel 3a smartphones running a modified version of AOSP 9.0 (The Android Open Source Project, or AOSP, is an open source branch of the Android operating system),[10] located in Australia. We modified the operating system by instrumenting the permission-checking APIs. What this means is that using our modified version of the operating system, whenever an unmodified Android app attempts to access a resource protected by Android's permissions system (`permission system'), our instrumentation makes note of this, so that we can understand which apps accessed protected user information during testing. This allows us to monitor the execution of individual apps without having to modify them. In testing Android apps, our instrumentation also monitored the payloads of network traffic, allowing us to examine even TLS-encrypted traffic. Unlike other app- and traffic-monitoring tools, our instrumentation was generally invisible to apps: we monitored apps' transmissions for the presence of information that indicated that they did not detect that they were running on "rooted" or otherwise modified/monitored devices.[11]

We tested each app a minimum of two times: first automatically (with a "robot" tester) and then with a real human using each app. Our testing procedure for Android was to first install an app by downloading its newest version from the Google Play Store or by "sideloading" it, which is the process of transferring an app from a computer via USB cable to the testing phone running our instrumentation. Once the app was installed, it was launched. The testing device was logged into a Google account associated with that device (as is the norm for Android). We first used a "robot" tester to automatically interact with the app (i.e., by performing random "clicks" and "swipes") while collecting data on the app's

[9]More details about our methodology may be found in several peer-reviewed publications that our team members have co-authored [10, 12, 9, 15, 14].

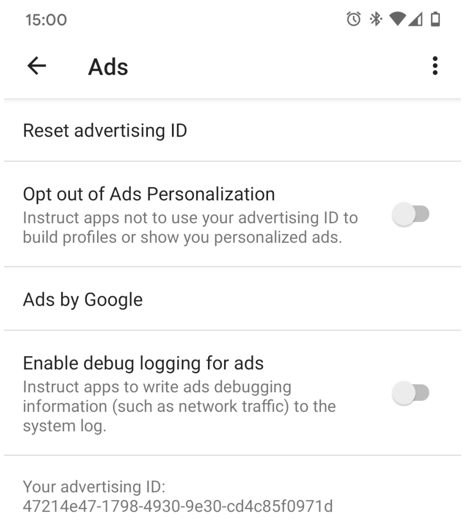
[10]While Android 10 ("Q") is the most recent version (as of this writing), estimated adoption rates remain at around 8% (i.e., around 92% of Android users are using version 9 or below). Thus, we performed our testing on Android 9, because at the time of testing, the vast majority of users were using either it or an even older one (with fewer privacy protections). Most carrier-locked devices only support system updates for a year or two after release, and therefore most Android devices run older versions. For this reason, we perform our testing on Android versions that most consumers are actually using, rather than the most recent version.

[11]We perform periodic testing so that when we do identify apps that detect that they are running on a modified device, we modify our instrumentation until it is no longer detected again.

behaviors while it was being used for a 10 minute time period. Finally, the logs generated during testing were downloaded from the phone for analysis. We then repeated the entire testing procedure, but instead of the automated "robot" tester, a human tested the app for an additional 10 minute time period.

Android offers users the option to opt out of tracking using a system-wide setting (Figure 6). When accessing the AAID, by policy, apps are required to check the status of this setting and then handle the AAID accordingly:

> *If reset, a new advertising identifier must not be connected to a previous advertising identifier or data derived from a previous advertising identifier without the explicit consent of the user. Also, you must abide by a user's "Opt out of Interest-based Advertising" or "Opt out of Ads Personalization" setting. If a user has enabled this setting, you may not use the advertising identifier for creating user profiles for advertising purposes or for targeting users with personalized advertising. Allowed activities include contextual advertising, frequency capping, conversion tracking, reporting and security and fraud detection."*[12]


15:00

Ads

Reset advertising ID

Opt out of Ads Personalization
Instruct apps not to use your advertising ID to build profiles or show you personalized ads.

Ads by Google

Enable debug logging for ads
Instruct apps to write ads debugging information (such as network traffic) to the system log.

Your advertising ID:
47214e47-1798-4930-9e30-cd4c85f0971d

Figure 6: System-wide ad tracking/personalization settings in Android.

In practice, this often means transmitting an additional parameter to indicate to data recipients that the user has opted out of tracking (in lieu of, e.g., using this setting to prevent apps from accessing the AAID altogether, which is how Apple's iOS treats a similar setting).[13] During testing, the tracking opt-out was *disabled* on all devices (i.e., apps were

[12]https://web.archive.org/web/20200528143805/https://play.google.com/about/monetization-ads/ads/

[13]https://web.archive.org/web/20191017062341/https://developer.apple.com/documentation/adsupport/asidentifiermanager/1614151-advertisingidentifier

not automatically instructed to disable tracking), reflecting the default setting, and therefore representing the experience of most consumers. While a considerable amount of research shows that consumers would much prefer to not be tracked,[14] paradoxically most consumers never change this setting because they likely do not know of its existence.[15,16]

[14]For example, Felt et al. found that 60% of study participants ($n = 3,115$) indicated that they would be "very upset" if they found out that a mobile app "used [their] phone's unique ID to track [them] across apps" [5].

[15]https://web.archive.org/web/20200628083057/https://adage.com/article/privacy-and-regulation/limit-ad-tracking-drops-ad-blocking-grows/303911

[16]For more information about consumers and the use of defaults, see the UK Competition and Markets Authority's recent report [13].

3 User Information

Background Information

How do you define "user information"?

We define "user information" as any information that describes an identifiable living person. However, when we use that term throughout this report, we specifically refer to the types of information described in this section (unless otherwise noted). User information can include profile and contact information, location information, as well as persistent identifiers.

What are persistent identifiers and why are they important?

An identifier is any piece of information that allows an individual—or device—to be uniquely identified. "Persistent" identifiers are identifiers that tend to not change over time. For example, motor vehicles have persistent identifiers in the form of license plates: a license plate uniquely identifies a vehicle and vehicles tend to have the same license plates over time. Thus, if someone records all of the license plates at a particular place over time, they can determine how many times in that period any individual vehicle was there. Similarly, if license plates are recorded at many different locations and that data is combined into a single dataset, one could use that to reconstruct the movements of individual vehicles in that dataset. As can be seen, combining a persistent identifier with information about where it was observed allows a data recipient to reconstruct an individual's activities. Using this knowledge, one could infer information about their routines, preferences, and demographics.

This is precisely how mobile tracking occurs. Mobile phones have various identifiers associated with them, including some that cannot be easily changed. As mobile phones are very personal devices, a unique identifier for a mobile phone is consequently a unique identifier for that individual, and can therefore be used to collect data about their activities, preferences, and demographics, simply based on data collection that associates it with the apps that were used, when, how, and where they were used.

Throughout the results section of this report, we refer to transmissions of user information with reference to a data type. In this section, we define each of these data types, describe what they are, what they look like, what purpose they serve, and the degree to which they can be easily changed by the user (e.g., to preserve privacy). For simplicity, we categorize these data types amongst three main categories: persistent identifiers that allow a user to be tracked over time and across services, sensitive user information that reveals a user's contact information or personal details, and fine-grained location data that allows a user to be physically located with a high degree of precision.

Persistent Identifiers

Persistent identifiers are unique numbers that allow an individual to be tracked over time across services. Some of them can be reset to prevent this type of tracking (e.g., AAID), whereas others are hardware-based and cannot be easily reset by the user (e.g., WiFi MAC).

AAID: This is the Android advertising identifier (sometimes known as the "Google advertising identifier" or GAID). It has the format of a Java-style UUID, such as `4e398b96-c1b7-4937-942c-1ab8d4e34b3a`. It is a resettable identifier, meaning that users can go through the system settings to reset it to a new value (this is the mobile tracking equivalent of clearing web browser cookies to prevent tracking across websites). Google's developer guidelines require that this is the only identifier that is used for behavioral advertising and other tracking/profiling purposes (see documentation).

Android ID: The Android ID is a semi-persistent identifier; it can be reset, but only through an extraordinary operation: a factory reset of the device. As such, the Android ID remains the same even when the AAID changes. Apps that send both together are therefore able to bridge changes to the AAID (i.e., resetting the AAID has no privacy-preserving effect). It has the format of a 16-digit hexadecimal number (e.g., `a553362398083b36`).

BT Name: This is the name of the Bluetooth interface associated with this device (i.e., the name that this device will appear as when other devices search for nearby Bluetooth devices). This is the human-friendly name, which is not guaranteed to be globally unique. This name may also uniquely identify a user.

GSF ID: The Google Services Framework Identifier ties the user's account to Google Services. It has the format of a 16-digit hexadecimal number (e.g., `37ee4f071b362be6`).

IMEI: This is the International Mobile Equipment Identity. This is used when connecting to cellular networks and to block stolen phones. It is a unique identifier that cannot be changed (it is illegal to change the IMEI in some jurisdictions). It does not change with factory resets and therefore remains the same even for refurbished mobile phones. It has the format of a 15-digit decimal number, such as `353626076259204`.

Serial #: This is the serial number of the phone. It is semi-persistent in that it requires a factory reset and a custom operating system to change. It has the format of a 16-digit hexadecimal number, such as `02597e74305f4802`.

SIM ID: The SIM ID is the identifier tied to the SIM card that is installed in the phone. As with the telephone number, the SIM ID can only change if the user removes the SIM card and installs a new one.

WiFi MAC: This is the hardware ("MAC") address of the phone's networking hardware. The MAC address is typically persistent, as it does not change with factory resets or other operations. It can be temporarily changed with non-trivial technical effort, but can only be persistently changed by rooting the phone. It has the format of 6 octets separated with a colon, such as `a8:b8:6e:4b:d0:a2`.

Sensitive User Information

We define *sensitive* user information to include personal contact information, demographic data, and other information that speaks directly to an individual's preferences.

Name: The full name associated with the Google Account that was provided when configuring the mobile phone. This corresponds to the owner of the phone.

Email Address: The email address that was used when first configuring the mobile phone. This corresponds to the owner of the phone's email address.

Phone #: This is the phone number associated with the SIM card that is installed on the phone. The phone number can change, but it requires a new SIM card in order to do so.

Package Dump: This refers to an app recording and transmitting the names of all of the other installed apps ("packages") on the device. This data is generally used for behavioral profiling purposes, as it can both be used to fingerprint a device and infer the user's interests based on the other apps that they use.

Username: This corresponds to the username of the user's app profile, which may be used to identify the user's profile.

User ID: This corresponds to a numerical identifier that identifies the user's app profile.

Location Information

Location information consists of either fine-grained GPS coordinates or other information that would allow the recipient to infer the user's location with a high degree of accuracy (e.g., WiFi SSID or WiFi BSSID). This does not include coarse-grained location data (e.g., city, state, or country) or data that would allow a recipient to infer the user's location on a coarse-grained level (e.g., IP address, which are included with every Internet transmission).

GPS Location: This means transmission of both the latitude and longitude in the same network transmission with more than 3 decimal places of accuracy, which is roughly within 100m. This can be considered a precise location, as it provides street-level accuracy.

BT SSID: This is the name of a nearby Bluetooth device. The SSID is the human-friendly name, which is not guaranteed to be globally unique. By determining the names of known nearby Bluetooth devices, a data recipient could use this information to infer the device's coarse-grained location.

BT BSSID: This is the MAC address of a nearby Bluetooth device and is a globally unique identifier. By identifying known nearby Bluetooth devices, a data recipient could use this information to infer the device's coarse-grained location.

WiFi SSID: This is the name of the WiFi router that the phone is either connected to or can perceive in the vicinity. The SSID is the human-friendly WiFi name, which is not guaranteed to be globally unique, although many routers have unique default SSIDs. The SSID does not uniquely identify a user, as it may change throughout the day and the router may have more than one user; it does, however, give coarse-grained location data of a comparable accuracy to GPS.

WiFi BSSID: This is the MAC address of the WiFi router and is a globally unique identifier. The BSSID does not uniquely identify a user, as it may change throughout the day and the router may have more than one user; it does, however, give coarse-grained location data of a comparable accuracy to GPS.

4 Analysis

For the purposes of this report, we tested the apps listed in Appendix E during June–July 2020 (the `testing period'). These apps were split across three categories, as determined by the ACCC: "Health" (103 apps), "Kids" (100 apps) and "Other" (797 apps). In this section, we present our analysis in terms of trends across these categories. We first present the types of user information that we observed being transmitted by each app during the testing period (e.g., location data, persistent identifiers, contact information, etc.), including the third-party software development kits (SDKs) that we identified, which provide insight into the *potential* third-party recipients for this data. Next, we provide an overview of the most common entities that we *actually* observed receiving user information from the apps that we tested during the testing period. We then present data on how apps made use of the permissions system in accessing sensitive data on the device during the testing period, including discussion of instances in which we observed the permission system being circumvented by several apps. Finally, we examine whether, during the testing period, in-app disclosures were presented to users, whether data was sent securely, and whether app developers made use of various privacy-related configuration options.

4.1 User Information Accessed and Transmitted

In this section, we examine the types of user information that apps transmitted during testing. This data reflects examples of actual data transmissions that we detected. Due to our testing methodology, we believe it reasonably represents practices a typical Australian consumer would experience during the course of their mobile device use. Nonetheless, many of the types of transmissions that we detected and documented are dynamic in nature: many data flows occur as a result of factors that are exogenous to our controlled testing environment and are often governed by stochastic processes. For example, the circumstances surrounding which ad networks are contacted to place an ad in a particular app, which advertiser wins an ad auction, or how in-app analytics functionality gets triggered may result in different data being sent to different third parties when running the same app multiple times. As a result, our observations from dynamic analysis represent the lower bounds of mobile app behavior: these are the data types and recipients that we *actually observed* during testing, but it is possible—and likely—that running these same apps again under different—or even the same—circumstances may yield additional data types flowing to additional recipients.

As a result, in this section we also examine apps' use of third-party Software Development Kits (SDKs) that we were able to identify using software analysis techniques during the testing period. Because SDKs are able to access the same user information as the apps that contain them, they represent the potential for user data to be shared with third parties: each SDK *could* transmit user information to the Internet servers belonging to that third-party SDK's parent company (or any other Internet servers, for that matter). Thus, while the transmissions that we observed during testing represent a lower bound for each app's privacy behaviors, analysis of their bundled SDKs allows us to understand additional *potential* third-party recipients of user information.

4.1.1 Data Types Transmitted

In this section, we examine the types of user information that was transmitted by apps in each category. As can be seen in Figure 7, the Android Advertising ID (AAID) was by far the most commonly transmitted type of user information.

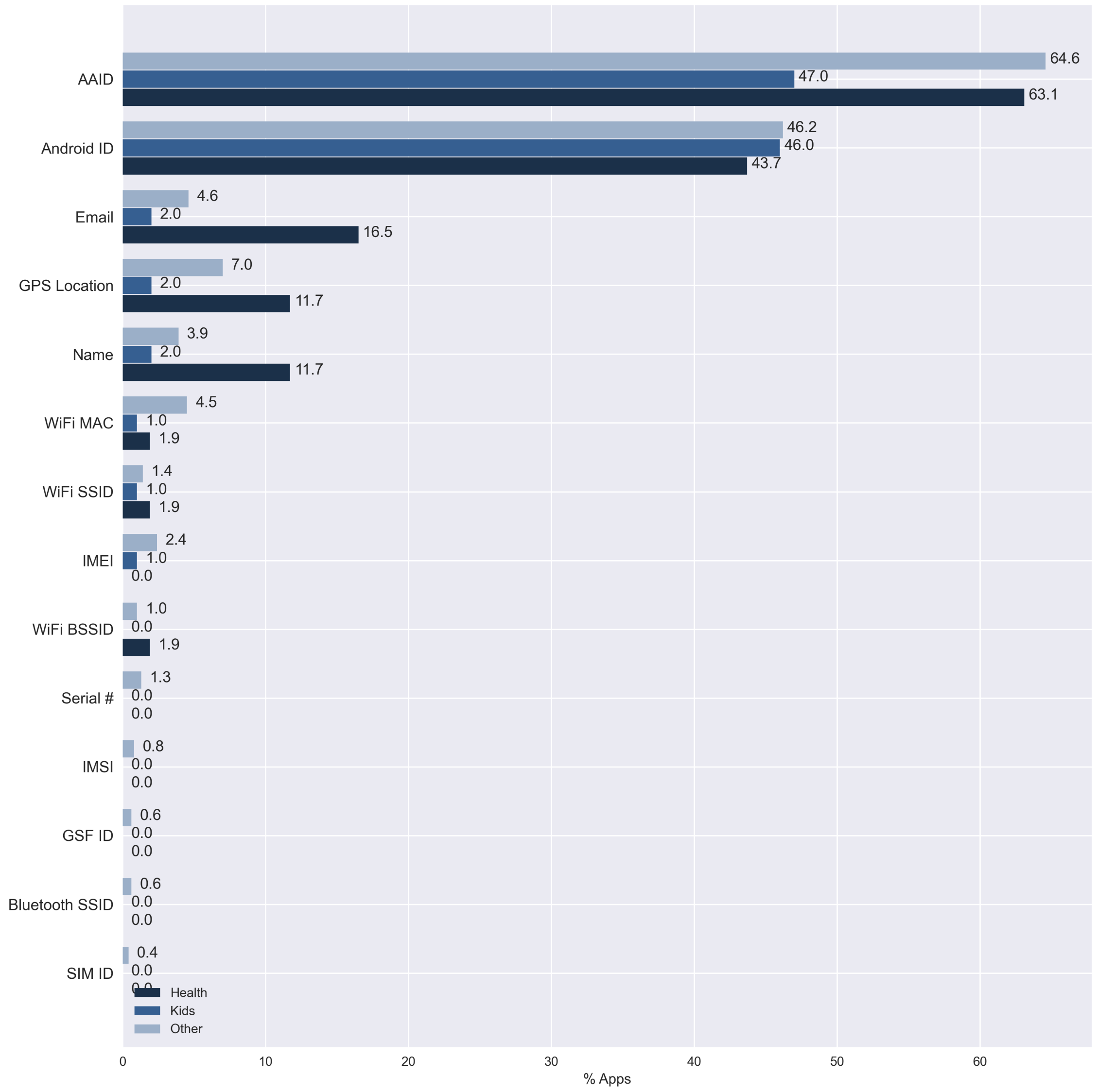
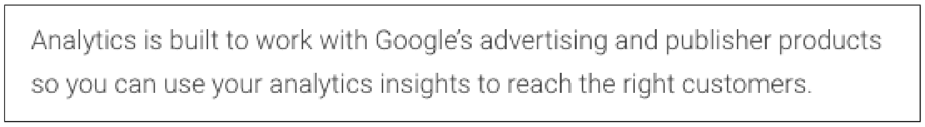
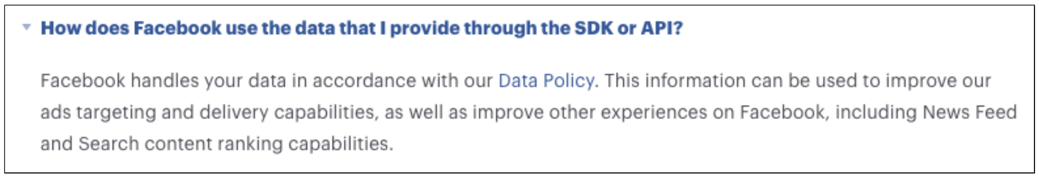


Figure 7: Percentage of apps in each category that transmitted various data types.

By policy,[17] the AAID is to be used only for advertising and analytics purposes, which means that when apps collect it, it is almost certainly being used for purposes that are secondary to providing consumers with core app functionality. While Google's policy makes exceptions for "contextual advertising, frequency capping, conversion tracking, reporting and security and fraud detection," from a technical standpoint, none of these activities *requires* the collection of a unique persistent identifier that can be used to track users across apps: all can be performed using either ephemeral session identifiers or identifiers that are unique for each app or developer (such as the Android ID starting in Android "O, " which was released in 2017).[18,19]

More to the point, the AAID "must only be used for advertising and user analytics,"[20] which are both secondary uses of user information. Further, much of the data collection under the guise of "analytics," which we discuss in Sections 4.1.3 and 4.2.1, is also in the service of advertising (e.g., better understanding app demographics in order to better target individuals with advertisements). For example (Figure 8), Google advertises that "Analytics is built to work with Google's advertising and publisher products so you can use your analytics insights to reach the right customers;"[21] Facebook discloses that data collected through its SDK "can be used to improve [their] ads targeting and delivery capabilities."[22] Therefore, by examining the collection of the AAID across the mobile app ecosystem, we can gain a sense of the amount of data collection that is being used for advertising (and potentially other) purposes.


Analytics is built to work with Google's advertising and publisher products so you can use your analytics insights to reach the right customers.


How does Facebook use the data that I provide through the SDK or API?

Facebook handles your data in accordance with our Data Policy. This information can be used to improve our ads targeting and delivery capabilities, as well as improve other experiences on Facebook, including News Feed and Search content ranking capabilities.

Figure 8: Google (above) advertises integrations between analytics and advertising products; Facebook (below) indicates that data collected through its SDK may be used for advertising purposes.

[17] `https://web.archive.org/web/20200528143805/https://play.google.com/about/monetization-ads/ads/`

[18] The Android ID (also known as the "SSAID"), is a persistent identifier that is readable by apps without special permissions. It is unique to the device and the app, which means that when accessed by an app, it will be a different value than when it is accessed by another app by a different developer. Thus, each app may use this identifier to identify users over time, but it was designed so that they—nor other third parties—cannot use it to identify users across different apps (i.e., each developer can only use this identifier to track users of *their* app).

[19] `https://web.archive.org/web/20200702180532/https://android-developers.googleblog.com/2017/04/changes-to-device-identifiers-in.html`

[20] *Ibid.*

[21] `https://web.archive.org/web/20200717211736/https://marketingplatform.google.com/about/analytics/`

[22] `https://web.archive.org/web/20200627212450/https://developers.facebook.com/docs/app-events/faq`

Among Health and Other apps, over 60% transmitted the AAID to various Internet servers during testing, whereas only 47% of Kids apps transmitted this identifier. Thus, a majority of the apps that we tested collected the AAID during the testing period, and thus enabled the ability to track individual consumers' behaviors over time, build detailed profiles about what apps they used, what they did within those apps, what ads they saw, and how they respond to those ads. This type of data is generally used to infer demographics and preferences for the purposes of serving highly-targeted ads (both within mobile apps, but also on websites and even in the physical world).[23,24,25]

We note that the percentage of apps transmitting the AAID was roughly two thirds less among Kids apps, and suspect that this is largely due to regulations in other jurisdictions that prohibit—or substantially limit—the collection of this type of data from minors. For example, both the E.U.'s GDPR and U.S.'s Children's Online Privacy Protection Act (COPPA) prohibit data collection from minors without verifiable parental consent. While we are not aware of any comparable laws in Australia, and certainly these laws do not apply to Australian consumers, *some* Australian consumers may be indirectly benefiting from them because many of the apps that they use are designed to also be used in these other markets. Nonetheless, almost half of all of the Kids apps transmitted the AAID.

Regarding the other types of data transmitted during the testing period, the next most prevalent was location data (in the form of precise GPS coordinates, GPS Location):[26] nearly 12% of Health apps and 7% of Other apps transmitted GPS Location, whereas 2% of Kids apps were observed transmitting location data at this level of precision. We should note that while the vast majority of apps did not collect GPS Location, several collected data that can be *used* to infer location with high granularity, such as WiFi BSSID or WiFi SSID. More importantly, not captured in this figure is that nearly every app transmitted data that can be used to infer location with coarser granularity: IP addresses, which are transmitted in the course of connecting to the Internet, can be used to identify individual Internet users with city-level accuracy. Therefore, every app that we observed transmitting any data automatically transmitted the user's IP address in the process.

[23]https://web.archive.org/web/20200717214750/https://www.consumerreports.org/privacy/google-sued-for-tracking-consumers-after-they-opted-out/

[24]https://web.archive.org/web/20200701132300/https://www.consumerreports.org/privacy/digital-billboards-are-tracking-you-and-they-want-you-to-see-their-ads/

[25]For example, Google's support documents note the ability to track consumers across devices (https://web.archive.org/web/20200106103215/https://support.google.com/analytics/answer/3234673?hl=en), while Facbeook's marketing materials promote a similar feature (https://web.archive.org/web/20191027080016/https://www.facebook.com/business/news/cross-device-measurement).

[26]We define "precise GPS coordinates" to mean within 100m of each testing phone's location.

Background Information

What are IP addresses and how are they used to determine location information?

Every device connected to the Internet has an "Internet Protocol" (IP) address: while a postal address is needed to send and receive mail, an IP address is needed to send and receive Information on the Internet. Thus, by nature of the Internet's design, whenever an app communicates with an Internet server, it reveals the IP address of the device on which it is running or that of the firewall it is behind. Databases exist that map these IP addresses to approximate geographical locations.[a]

[a] https://en.wikipedia.org/wiki/Geolocation_software

Finally, the next most prevalent type of user information that we observed being collected during testing was the WiFi MAC. Among Health and Other apps, 1.9% and 4.5% transmitted the WiFi MAC, respectively, whereas we observed 1% of Kids apps transmit it during testing. While this proportion may seem low, we note that starting in Android 6.0,[27] which was released in 2015, Google began prohibiting programmatic access to the WiFi MAC for privacy reasons: it was being used by third party tracking companies as a persistent identifier, and because it is based in hardware, it cannot easily be changed by consumers (i.e., there is no way of doing the analogue of clearing web browser cookies). For this type of tracking, Google requires that only the AAID be used, which was unveiled at this same time that access to the WiFi MAC was prohibited. Nonetheless, our team has previously documented various security vulnerabilities that exist in Android, that when exploited, allow apps to collect data by circumventing the permission systems' protections [10]. In conducting the testing for this report, we identified new "vulnerabilities" that several apps were "exploiting,"[28] potentially violating Google's policies and circumventing Android's privacy protections, which we document in Section 4.4.

[27] https://web.archive.org/web/20200510021708/https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html

[28] https://en.wikipedia.org/wiki/Vulnerability_(computing)

Example

Collection of WiFi Router Information

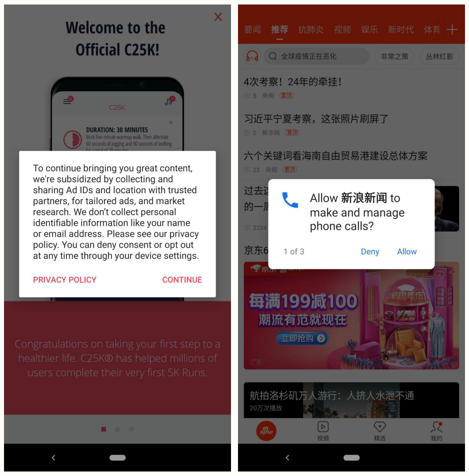
Some apps collect information about a user's environment, like the SSID (WiFi SSID) and BSSID (WiFi BSSID) of the connected WiFi router. The SSID refers to the name of the WiFi network, whereas the BSSID refers to its globally-unique MAC address (similar to how a mobile device's WiFi MAC is a globally-unique identifier that can be used as a "serial number" to identify it). Because WiFi routers tend to be in fixed locations, databases exist to map these WiFi identifiers to fairly precise GPS coordinates.[a] While in theory this information can be used to provide more accurate geolocation data to apps and services, it is also an identifier that can be used to learn about a user, their devices, their habits, and even their acquaintances. For example, a data recipient who receives data from multiple devices sharing the same WiFi BSSID knows that those devices are connected to the same WiFi network, and therefore are in physical proximity. If the data recipient also knows that the WiFi BSSID corresponds to a WiFi router located in a residential area, they may infer that those devices belong to individuals within the same household. If the data recipient knows that the WiFi BSSID corresponds to a workplace, they may infer that those individuals are employed there. Through the collection of this data type, data recipients could obtain data of commercial value, such as inferences about an individual's religious faith, shopping habits, or favorite hotel chains and restaurants, whenever that individual connects to a WiFi network.

Category	WiFi SSID	WiFi BSSID
Health	2	2
Kids	1	0
Other	11	8

In the **Health** category, "C25K® - 5K Running Trainer" (com.c25k; 1,000,000 installs), developed by Zen Labs Fitness, was observed during the testing period collecting and transmitting both of these identifiers. This app allows users to track their progress toward fitness goals, and so one can imagine many appropriate use cases that would require access to high-accuracy location data (GPS Location) through the `ACCESS_FINE_LOCATION` permission. However, during our testing, the data was collected by third-party libraries and sent to their servers, meaning that these libraries are availing themselves of data protected by a location permission that the user has granted to the app.

During our testing of this app, and the two other examples that we describe below, we observed router information being sent to the hostnames depicted in the following table:

App	Data Type	Destination
"C25K® - 5K Running Trainer" (com.c25k; 1,000,000 installs), developed by Zen Labs Fitness	WiFi BSSID	api.smartechmetrics.com
	WiFi SSID	api.smartechmetrics.com
	WiFi BSSID	api.myendpoint.io
	WiFi SSID	api.myendpoint.io
"Weatherzone" (au.com.weatherzone.android.weatherzonefr...; 1,000,000 installs), developed by Weatherzone	WiFi BSSID	control.kochava.com
	WiFi SSID	control.kochava.com
"新浪新□" (com.sina.news; 100,000 installs), developed by Sina.com	WiFi BSSID	beacon.sina.com.cn
	WiFi SSID	beacon.sina.com.cn
	WiFi BSSID	abroad.apilocate.amap.com
	WiFi SSID	abroad.apilocate.amap.com


Welcome to the Official C25K!
C25K
DURATION: 30 MINUTES
要闻 推荐 抗肺炎 视频 娱乐 新时代 体育
全球疫情正在恶化
非常之策
丛林红影
4次考察！24年的牵挂！
习近平宁夏考察，这张照片刷屏了
六个关键词看海南自由贸易港建设总体方案
每满199减100
潮流有范就现在
立即抢购
航拍洛杉矶万人游行：人挤人水泄不通
20万次播放
视频
精选
我的

To continue bringing you great content, we're subsidized by collecting and sharing Ad IDs and location with trusted partners, for tailored ads, and market research. We don't collect personal identifiable information like your name or email address. Please see our privacy policy. You can deny consent or opt out at any time through your device settings.

PRIVACY POLICY CONTINUE

Congratulations on taking your first step to a healthier life. C25K® has helped millions of users complete their very first 5K Runs.

Allow 新浪新闻 to make and manage phone calls?

1 of 3 Deny Allow

Though pinpointing the owners of `smartechmetrics.com` and `myendpoint.io` is difficult, a privacy-policy analysis by Dehaye and Reardon suggests that they belong to X-Mode [2, 1], an organization that specializes in harvesting and repurposing location data for uses including public health and law enforcement.[b] And, while *C25K®* was observed to have an in-app dialog at the outset which described its data-collection and monetization strategy, from the information provided, our view is that it would be difficult for the typical user to know (without their own extensive research) the specific identifiers that are collected, where they are sent, who ultimately receives them, what they do with them, and the overall implications of this data being collected to begin with.

In the **Other** category, two examples of apps transmitting router information during our testing are "Weatherzone" (au.com.weatherzone.android.weatherzonefr...; 1,000,000 installs), developed by Weatherzone and "新浪新□" (com.sina.news; 100,000 installs), developed by Sina.com. The former, which claims to provide users with weather information, transmitted WiFi router information to Kochava, a company that provides mobile advertising services. The latter app was observed collecting and transmitting both of these identifiers as well, though we also observed it transmitting the IMEI, IMSI, Serial #, WiFi MAC, GSF ID, SIM ID, and GPS Location to 7 distinct endpoints across 4 domains: *sina.cn, amap.com, sina.com.cn*, and *qchannel03.cn*. Occasionally, there are reasons to capture this breadth of data, much of which is restricted by privacy-conscious policies for Google's Play Store (e.g., the collection of WiFi MAC has been prohibited since 2015).[c] However, it is not clear to us why this particular app needed to access permanent and unique identifiers, such as the IMEI and SIM ID—both of which can be used to link users and their devices across services, circumventing system-wide privacy controls.

[a]https://en.wikipedia.org/wiki/Wi-Fi_positioning_system

[b]https://arxiv.org/pdf/2006.10719.pdf

[c]https://web.archive.org/web/20200510021708/https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html

4.1.2 SDKs Identified

Background Information

Software Development Kits (SDKs)

Developers bundle third-party Software Development Kits (SDKs) within their apps to save themselves from having to develop that particular functionality themselves. For example, rather than creating bespoke functionality to convert GPS coordinates into cities or counties, an app developer may integrate an SDK with such functionality. Some SDKs support primary app functionality and are therefore necessary—from the user's perspective—whereas others are for secondary purposes, such as advertising and analytics. For example, an analytics SDK could provide the app developer with a means of instrumenting their app to understand what users do while using it, as well as identifying those users to understand their demographics (e.g., including what other apps they use). A debugging or error-reporting SDK could allow an app to collect and transmit information about app crashes. An advertising SDK could provide the app developer with a way to monetize their app by showing highly-targeted advertisements to users (both in the app, as well as when they are identified in the future when using other apps and websites that share data with the same advertising networks).

Third-party SDKs have many uses, however, some may come with privacy implications. For example, some SDK providers integrate development support capabilities, like game engines and crash reporting, in SDKs that also provide support for secondary purposes, like advertising and analytics. This is the case with the Unity 3D and Google Firebase SDKs, respectively. SDKs, therefore, should be used with consideration and caution. In many cases, an app developer may integrate an SDK to fulfill a particular need, but not realize that the SDK is also collecting data for secondary purposes (e.g., the app developer failed to configure it in a certain way, the SDK deceived the app developer, etc.), such as behavioral profiling for advertising purposes. Inconsistent SDK configurations, therefore, have the potential to contradict what is disclosed in an app's privacy disclosures to users. For example, an app developer may not think that their app is sharing location data with third parties, and therefore says so in its privacy policy, but due to an SDK misconfiguration, users' location data ends up being shared with an advertising network.

The presence of third-party SDKs within apps may signal the potential for additional data collection that was not directly observed during our testing. Due to the way that the Android operating system is designed, the permissions system makes no distinction between whether protected data is being accessed by an app or an SDK embedded within that app. That is, third-party SDKs have access to all of the same user information accessible to their host apps, and therefore they could potentially be used to transmit this information to third parties. Therefore, the number of SDKs likely to transmit user information found within an app is a reasonable measure of its data-sharing potential.

We identified 407 different SDKs embedded within the 1000 apps that we tested. Appendices B.3, C.3, and D list the 40 most common SDKs, along with the number of apps in which they were detected among Health, Kids, and Other apps, respectively. Figure 9 depicts the prevalence of the 10 most popular SDKs among each of the three categories.

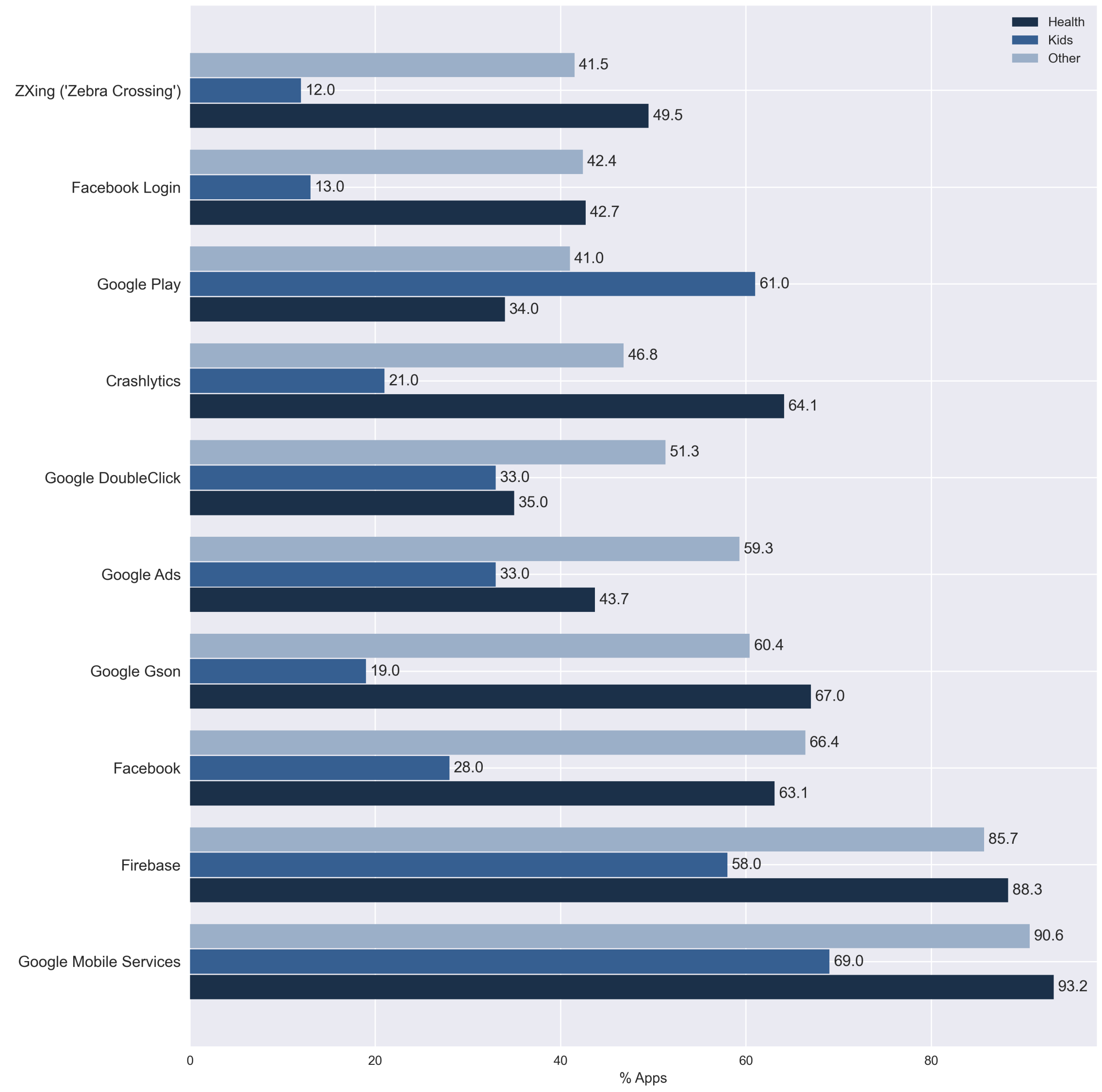


Figure 9: The percentage of apps containing the 10 most popular SDKs.

As can be seen, the most prevalent SDKs are development tools that provide various programming frameworks. For example, Google Mobile Services is in nearly every app because it is tightly integrated with the Android framework, as is Google's Firebase platform (a newer product, and therefore unlikely to appear in older apps that have not been updated). Many other SDKs are unlikely to transmit user information, because they do not require Internet access: they support developers by providing programming frameworks and libraries to speed up software development. On the other hand, some SDKs do require Internet access. For example, the Facebook Login SDK, a component of the broader Facebook SDK that allows users to login to apps with their Facebook accounts,[29] may transmit users' information in the service of providing required functionality. Other SDKs that require Internet connectivity, such as the Facebook Analytics SDK, which allows app developers to collect information about how users use their apps, collect users' information for purposes that are secondary to providing expected app functionality.

Table 1: The 10 most common advertising and analytics SDKs identified across all apps.

SDK Name	Parent Company	Category	# of Apps	(% of Apps)
Google Mobile Services	Google	Development Aid Advertisement Mobile Analytics Authentication	910	(91.0%)
Firebase	Google	Development Aid Mobile Analytics Advertisement LBS Utility	832	(83.2%)
Facebook	Facebook	Social Network Advertisement Mobile Analytics	622	(62.2%)
Google Ads	Google	Advertisement	551	(55.1%)
Google DoubleClick	Google	Advertisement	478	(47.8%)
Crashlytics	Google	Mobile Analytics	460	(46.0%)
Facebook Login	Facebook	Digital Identity Advertisement Mobile Analytics	395	(39.5%)
Facebook Share	Facebook	Social Network Advertisement Mobile Analytics	381	(38.1%)
Facebook Analytics	Facebook	Mobile Analytics	362	(36.2%)
Facebook Ads	Facebook	Advertisement	320	(32.0%)

[29]App developers are incentivized to embed the Facebook Login SDK because it saves them the effort of implementing their own authentication system for their app and it allows them to learn more information about their users (Facebook allows them to access those users' information).

Because we are primarily concerned with SDKs that are most likely to transmit sensitive user information for secondary purposes, we focus on those that we label as "Advertisement" or "Analytics,"[30] services that require Internet access and are often associated with the collection of user information. Table 1 depicts the 10 most common SDKs among these.

One caveat when identifying SDKs within Android apps is that due to obfuscation of either the app or SDK code, we may not be able to positively identify a particular SDK. Similarly, we may be able to identify the third-party SDK developer, but we cannot positively identify which of its multiple services is present. For example, Facebook's SDK offers multiple services:[31] Facebook Login so app users can login with their Facebook accounts; Facebook Analytics to help developers track their users; Facebook Social so users can post in-app events to their Facebook profiles; and so on. In some cases, we identified which of these Facebook services an app was using, whereas in other cases we labelled the SDK simply as "Facebook" because we were unable to disambiguate it. Relatedly, SDKs may get renamed or rebranded over time. For example, "Crashlytics" was once an independent company,[32] before it was acquired by Twitter in 2013, and then by Google in 2017 and rebranded as being part of Google's Firebase platform (itself a 2014 Google acquisition).[33]

Because apps often do not contain the most recent SDK versions (e.g., a 2017 study of over 1M apps found that nearly all were using outdated SDKs [3]), we instead present these results in Table 2 in terms of the parent organizations who are currently—to the best of our knowledge—responsible for the SDKs that we identified.

Table 2: The top 10 parent companies of identified advertising and analytics SDKs.

Parent Company	**# of Apps**	**(% of Apps)**
Google	910	(91.0%)
Facebook	622	(62.2%)
Oracle	236	(23.6%)
Unity Technologies	227	(22.7%)
AppsFlyer	183	(18.3%)
Twitter	177	(17.7%)
Integral Ad Science Inc.	131	(13.1%)
AppLovin	128	(12.8%)
IronSource	121	(12.1%)
Vungle	120	(12.0%)

[30] Firebase is classified as both a Development Aid and an Analytics and Advertisement SDK because it offers analytics functionality that involves collecting user information that may be used for advertising purposes.

[31] `https://web.archive.org/web/20200423125122/https://developers.facebook.com/products/`

[32] `https://en.wikipedia.org/wiki/Crashlytics`

[33] `https://en.wikipedia.org/wiki/Firebase`

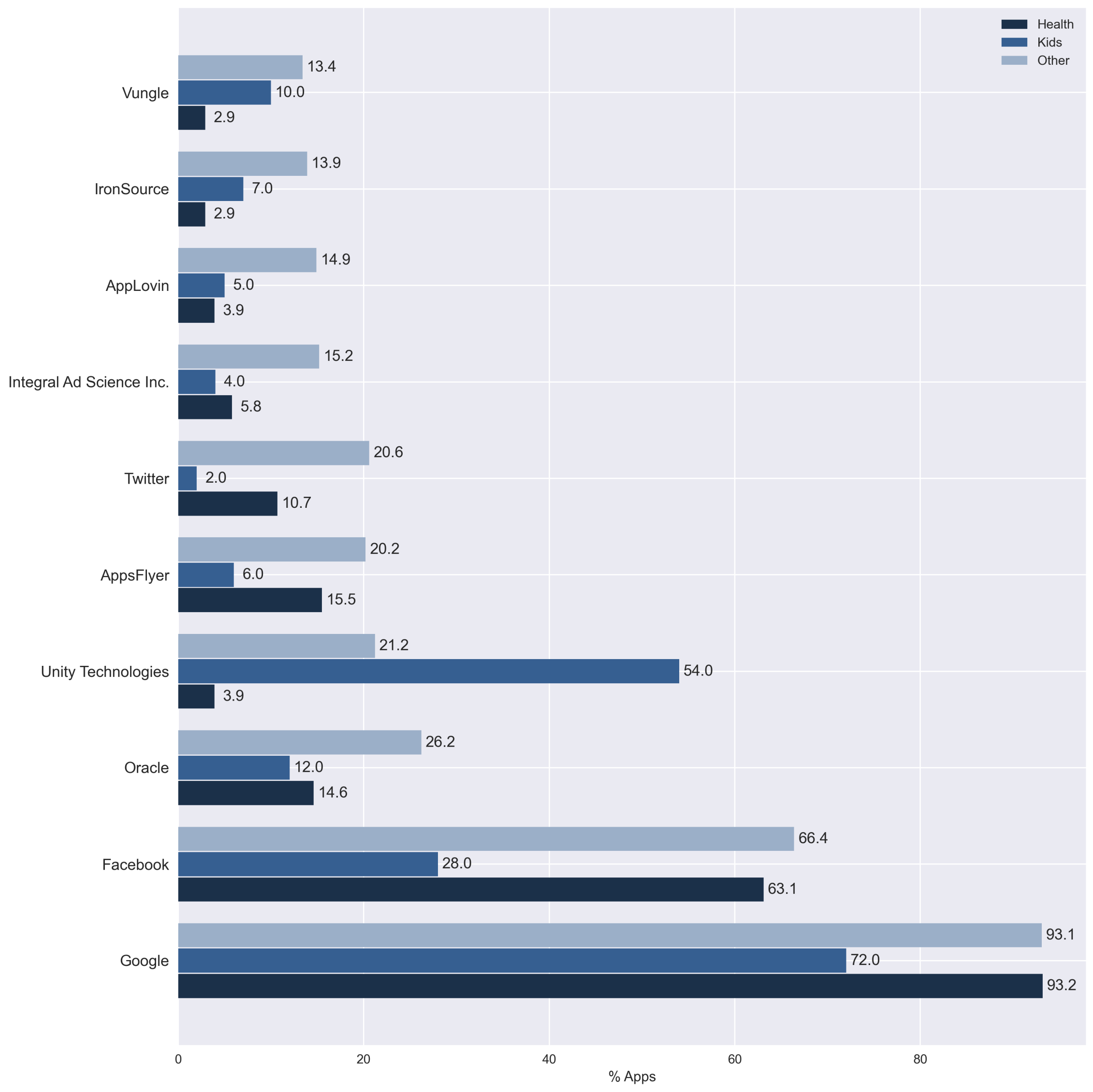


Figure 10: The percentage of apps observed containing one or more Advertisement or Analytics SDKs from the 10 most prevalent parent companies.

Figure 10 depicts the prevalence of the 10 most popular Advertisement and Analytics SDK parent organizations among each of the three app categories. Overall, our results show that Google has the potential to collect data from nearly all Android apps. This is far from surprising: Android is their platform, and therefore many apps contain their SDKs, because they offer a broad range of functionality.

Moreover, every Android device sold by an OEM-certified vendor comes with several Google-proprietary components pre-installed. While this report documents the user information transmitted to various third parties by individual Android apps that we tested during the testing period, this ignores the user information that we observed Google collecting from the operating system itself using its various proprietary components. For example, one of those proprietary components is Google Play Services:[34] every time an Android user installs an app from the Play Store, Google receives a notification reporting the app installation event, along with user information extracted from the device, including the AAID, as shown in Figure 11.

```
POST /play/log?format=raw&proto_v2=true HTTP/1.1
Content-Encoding: gzip
Content-Type: application/x-gzip
User-Agent: Android-Finsky/20.4.18-all [0] [PR] XXXXX1
Authorization: Bearer XXXXX2
Host: play.googleapis.com
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Length: 1901


[...]
aosp_XXXXX3.Googler.Androidz.sargo...
[...]
..... ......*............42_"7.....com.fanatee.cody..delivered_organic..
[...]
:.82041800.........*$a3fAAID.........b.........2..........
[...]
```

Figure 11: Example (truncated and redacted) of an encrypted protobuffer transmission from Google Play Services (package name: `com.google.android.gms`) to `play.googleapis.com`. Observe the value reporting an app installation event (com.fanatee.cody) and the AAID (redacted as a3fAAID).

For the vast majority of Android devices, Google automatically receives information about the apps that individual users choose to install on their devices. While this information could be used for providing statistics to app developers on their Google Play console or for obtaining telemetry and intelligence for the Google Play Protect program, it could be potentially used for advertising purposes. In fact, Google Analytics[35] and Google's Firebase documentation[36] suggest that Google Play Services are fundamental to enabling Google's analytics and advertising services on Android: developers must declare in their Android software project an explicit dependency on Google Play Services to integrate any of these SDKs in their products. So while our testing results indicate that a large number of apps communicate directly with Google, this underreports the fact that Google directly receives information *about* every app installed through the Google Play Store.

[34] https://web.archive.org/web/20200716131920/https://www.android.com/certified/

[35] https://web.archive.org/web/20200718105348/https://developers.google.com/analytics/devguides/collection/android/v4/

[36] https://web.archive.org/web/20200427152113/https://developers.google.com/android/guides/google-services-plugin

We identified Facebook's SDK in over 62% of all apps that we examined: 28% of Kids apps and over 60% of all other apps. By default (i.e., without configuration by the app developer), an app bundling Facebook's SDK will report to Facebook's servers that it was installed by this specific user, as well as how long and often the app is used by that user, including when they make in-app purchases.[37] This information is associated with users' persistent identifiers, unless the app developers specifically disables this behavior (or the user enables the system-wide tracking opt-out). Specifically, this means that almost two thirds of the apps that we examined have the ability to transmit users' information to Facebook, regardless of whether or not those users have Facebook accounts. According to Facebook's privacy policy, this information may be used by Facebook—in concert with user information that it has obtained from other sources (including other apps, websites, and third parties)—to then profile these users so that they can later be targeted with advertisements.[38] For example, from their privacy policy:

> *"We collect information from and about the computers, phones, connected TVs and other web-connected devices you use that integrate with our Products,*[39] *and we combine this information across different devices you use. For example, we use information collected about your use of our Products on your phone to better personalize the content (including ads) or features you see when you use our Products on another device, such as your laptop or tablet, or to measure whether you took an action in response to an ad we showed you on your phone on a different device."*[40]

Beyond Facebook, the next most prevalent company was Oracle, which was associated with an SDK found in almost 25% of all apps. Oracle is the parent company of Moat, which tracks what users do within apps in order to provide analytics services, including measuring the effectiveness of ad campaigns.[41]

Unity, like Google and Facebook, offers a variety of services to developers that include development tools, as well as analytics and advertising functionality.[42] Thus, they are able to collect data from over a fifth of the mobile apps that we analyzed, though surprisingly are in a disproportionate number of Kids apps (i.e., over half). Some of this data, we have previously noted [10], is collected using methods that circumvent the permissions system. Of the rest of the companies in the top 10, all are either in the advertising business directly or provide data and services to those who are.

[37] `https://web.archive.org/web/20200614223129/https://developers.facebook.com/docs/marketing-api/app-event-api`

[38] `https://web.archive.org/web/20200614050621/https://www.facebook.com/policy/`

[39] Facebook defines "Products" as including their SDK (`https://web.archive.org/web/20200619080733if_/https://www.facebook.com/help/1561485474074139?ref=dp`): "The Facebook Products also include Facebook Business Tools, which are tools used by website owners and publishers, app developers, business partners (including advertisers) and their customers to support business services and exchange information with Facebook, such as social plugins (like the "Like" or "Share" button) and our SDKs and APIs."

[40] *Ibid.*

[41] `https://moat.com/`

[42] `https://unity.com/`

4.1.3 SDK Privacy-Related Configuration Options

Many third-party Software Development Kits (SDKs) that collect user information from users of mobile apps offer app developers various privacy configuration options. In some cases, these options allow the app developer to limit how the SDK collects and shares users' user information with its parent company and other third parties. In other cases, these options allow the app developer to disable data collection until only after the user has explicitly consented. Sometimes these options are necessary to satisfy various privacy regulations, such as the General Data Protection Regulation (GDPR) in Europe or the California Consumer Privacy Act (CCPA) in the U.S. state of California. Nonetheless, regardless of whether or not the use of these types of options is mandated in all regions (e.g., we are not aware that using these privacy configuration options would be *required* for an app serving Australian consumers), their use is good practice, and reflects developers taking steps to respect their users' privacy. Thus, we believe that examining the prevalence of their usage across the apps that we tested is a reasonable heuristic for measuring developers' awareness of these privacy-related configuration options and their willingness to use them. At the same time, because the use of these options often determines whether or not a data recipient will automatically use the received data—often transmitted in the context of "analytics services"—for advertising purposes, it also serves as a measure of the data sources that power various advertising companies.

We examined the privacy-related configuration options for some of the most popular analytics and advertising SDKs that were found across all of the apps. For each SDK, we describe the privacy-related configuration options that we were able to identify based on that SDK's public documentation.[43] We performed static analysis on each app to detect the presence of several of these options. One of the limitations of this approach is that it may yield both false positives and negatives. For example, some of these errors may be due to bundled legacy SDKs that are no longer used by an app, if the SDK documentation only describes the options for newer versions of an SDK, or if code that impacts privacy-related behaviors is loaded at runtime. Nonetheless, we have no reason to believe that these types of errors are more or less likely among any particular type of app, and therefore we assume that errors in our static analysis are equally distributed across the app categories that we examined. Thus, we believe that this is a useful metric for comparing across categories of apps.

4.1.3.1 Google Ads / DoubleClick By default, when integrated into an Android app, the Google Ads SDK (which integrates functionality from DoubleClick) will collect user information and use it for it Google's advertising purposes. However, the SDK can be configured so that instead of immediately collecting this data, it will only do so after the user has explicitly consented to it. App developers may be required to use these configuration options when

[43]We were unable to conduct this analysis for all of the SDKs that we identified above, as in many cases we were unable to locate documentation about the privacy options that they offer. Similarly, this analysis is meant to examine trends amongst the most popular SDKs and is not meant to be an exhaustive or representative survey of the ecosystem.

making their mobile apps available to consumers in certain jurisdictions, in order to comply with various regulations. For example, the Google Ads SDK provides information about configuring the SDK to support gaining consent from European users for GDPR purposes.[44] Although these methods center around obtaining consent prior to data collection for behavioral advertising purposes, they include instructions on how to use the Consent SDK and integrate its information with Google's other SDKs that an app developer may bundle with an app (e.g., Google Ads). The prevalence of the Consent SDK among our dataset may serve as an indicator of (1) app developers' awareness of their own compliance obligations and (2) the scope of Google's data collection from mobile apps for advertising purposes that occurs *by default*. Examining all of the apps in our dataset, we observed evidence of the Consent SDK within a total of 63 apps. We observed references to the "`ConsentInfoUpdateListener`" function—we excluded files belonging to the SDK itself—in 52 apps, suggesting that these apps are making use of this functionality to allow the user to consent to data collection.[45] This corresponded to 16 Health apps and 36 Other apps.

The Google Ads SDK also offers support for disabling data collection if the user is under the age of consent. For example, one would expect child-directed apps that bundle Google's Ads SDK to use this functionality. We detected usage of this function by scanning each app for references to the "`setTagForUnderAgeOfConsent`" function and found it within 296 apps: 28 Health apps, 14 Kids apps, and 254 Other apps.

4.1.3.2 Google Crashlytics (Firebase) Crashlytics is a crash-reporting tool that is now part of the Google Firebase suite of software development tools. It allows developers to collect information about the functioning of their apps, such as any error logs or crash reports. This can help developers improve the performance of their apps or diagnose software bugs. Because this data can include sensitive information (e.g., the contents of device memory, which may contain passwords or user information), Google offers developers a configuration option (i.e., setting "`firebase_crashlytics_collection_enabled`" to "`false`" within the `AndroidManifest.xml` file of the app) to disable data collection by default, so that users of the app are given an opportunity to first consent to this type of data collection.[46] That is, when this configuration option is not present, Crashlytics data collection will occur automatically, without user consent. While the documentation describes this functionality in the context of the European Union's General Data Protection Regulation (GDPR), the privacy protections offered by this setting are relevant to consumers worldwide.

We examined the number of apps that followed Google's advice to gain user consent prior to collecting data using Crashlytics. We scanned all of the apps for the existence of "`firebase_crashlytics_collection_enabled`" set to "`false`" within each app's `AndroidManifest.xml`. Overall, across all 1000 apps tested, we observed 50 apps using this

[44] `https://web.archive.org/web/20200710123205/https://developers.google.com/admob/android/eu-consent`

[45] Though this number may include false positives due to the integration of the Consent SDK within other third-party SDKs within an app.

[46] `https://web.archive.org/web/20200605035738/https://firebase.google.com/docs/crashlytics/customize-crash-reports`

configuration option to disable the automatic collection of user information. Across the three categories, this corresponded to 7 Health apps (11% of those that we identified as containing the Crashlytics SDK) and 43 Other apps (12% of those containing Crashlytics), while none of the 21 Kids apps containing Crashlytics appeared to include this configuration option. Overall, this suggests that few apps are heeding Google's advice to attain user consent prior to transmitting user information to Google's servers.

4.1.3.3 Facebook As noted earlier, Facebook's SDK provides a wide range of functionality, including performing data collection for analytics and advertising purposes. Its documentation indicates that users' user information will be collected by default, as soon as the app is run, unless the app developer either chooses to disable automatic data collection (i.e., by setting "`com.facebook.sdk.AutoLogAppEventsEnabled`" to "`false`" within the `AndroidManifest.xml` file of the app) or by preventing the SDK from automatically initializing altogether (i.e., by setting "`com.facebook.sdk.AutoInitEnabled`" to "`false`" within the `AndroidManifest.xml` file).[47] The documentation instructs app developers to use these functions to respect users' privacy rights by only turning on data collecting after the user has provided their consent to do so.

We examined the number of apps that disabled Facebook's automatic data collection by setting either "`AutoLogAppEventsEnabled`" or "`AutoInitEnabled`" to "`false`" within the `AndroidManifest.xml` file of the app. Of the 109 apps that were identified as using one of these configuration options (across all 622 apps that included any of Facebook's SDKs), 8 were Health apps (12% of 65 Health apps that contained the Facebook SDK), 1 was a Kids app (4% of 28 Kids apps with the Facebook SDK), and the remaining 90 were Other apps (18% of 529 Others apps with the Facebook SDK). As before, this finding suggests that the vast majority of apps containing Facebook's SDK transmit user information to Facebook's servers by default, without first attaining users' explicit consent.

[47] `https://web.archive.org/web/20200710151530/https://developers.facebook.com/docs/app-events/gdpr-compliance/`

4.2 Third-Party Data Recipients

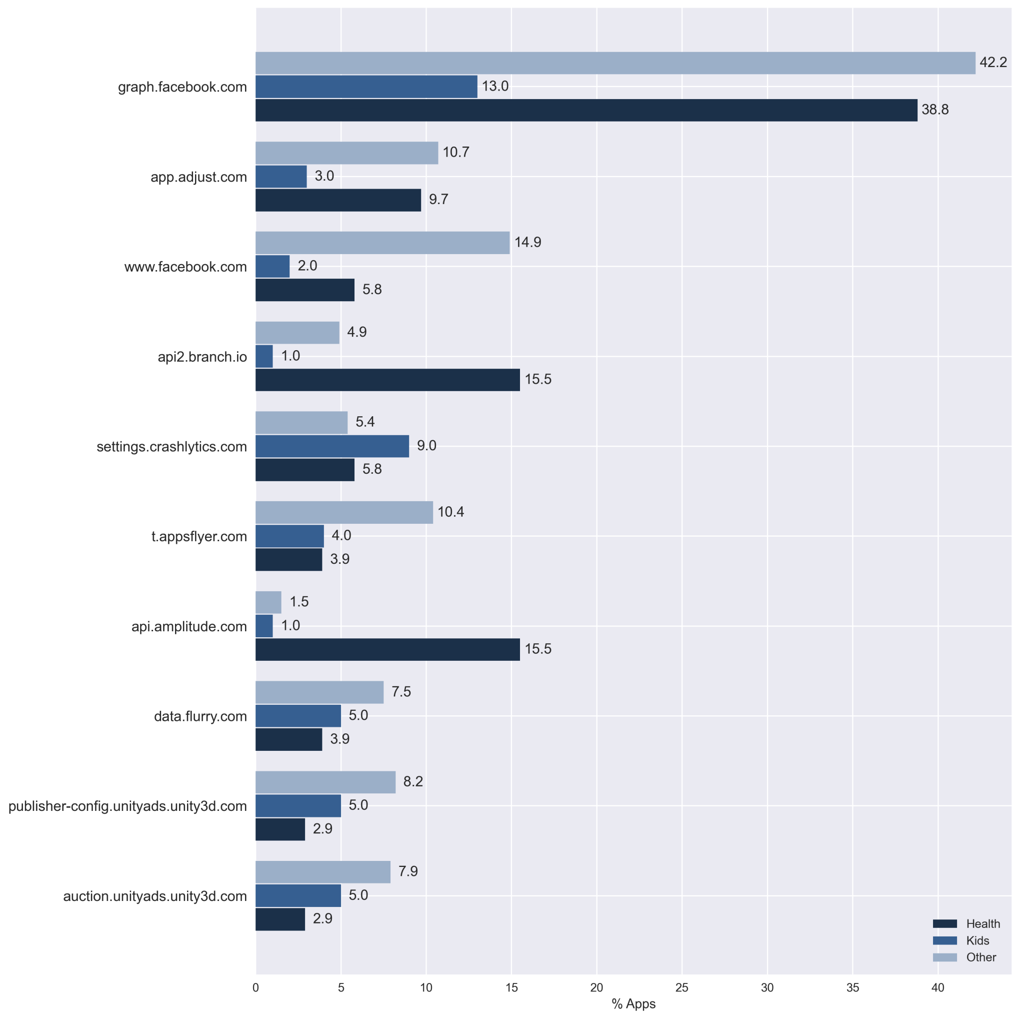


Figure 12: Percentage of apps sending user information to top 10 third-party hostnames.

In the previous section, we examined the types of data that apps transmitted, as well as the SDKs that they embed, which provides an estimate of the number of *potential* data recipients. In this section, we examine the entities who we observed actually receiving user information from mobile apps during the testing period, which we consider to be a lower bound for the privacy risks to which normal consumers may be exposed. This serves as a contrast: while the prior section describes potential privacy risks, this section provides a snapshot of observed practical privacy behaviors, rather than theoretical ones. Figure 12 shows a plot of the top third-party data recipients (by hostname) that received user information from apps across categories during testing.

As can be seen in the chart, `graph.facebook.com`, the primary API endpoint for Facebook's SDK is far and away the most common recipient of user information during our testing. A total of 38.9% of all apps tested transmitted some type of user information—of those described in Section 3—to this address. However, this is only one of Facebook's hostnames that we observed receiving data. Separately, we observed connections to `www.facebook.com`, which appear to be due to a variant of the normal SDK, though transmitting similar data. As a result, viewing data recipients based on corporate entities receiving the data, rather than the hostnames of the API endpoints, may be more instructive. Figure 13 shows that during the testing period, Facebook was the top data recipient (during testing, 40.5% of all apps transmitted user information to any destinations that we identified as being owned by Facebook).

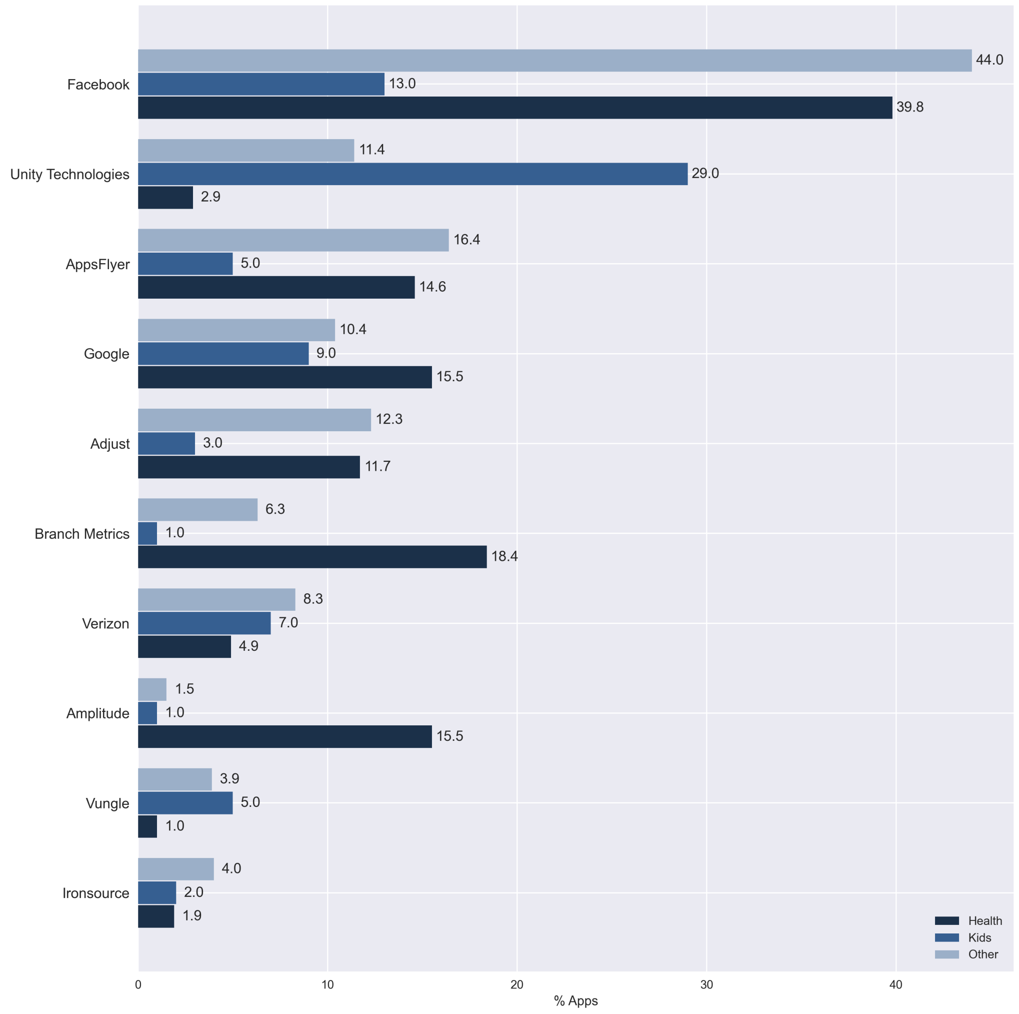


Figure 13: Percentage of apps sending user information to the top 10 recipient companies.

As the rates of SDK integration predicted, user information observed being transmitted to Facebook by Kids apps (13.0%) was much less likely than compared to Health apps (39.8%) or Other apps (44.0%). Comparing with Figure 10, we can see that across the three categories, the presence of the Facebook SDK in an app resulted in transmissions of user information to Facebook's servers roughly 50–80% of the time. Overall, the top 10 recipients of user information from apps can be seen in Table 3.[48]

Table 3: The top 10 recipients of user information across all apps during testing.

Parent Company	**# of Apps**	**% of Apps**
Facebook	405	40.5%
AppsFlyer	151	15.1%
Unity Technologies	123	12.3%
Adjust	113	11.3%
Google	108	10.8%
Twitter	81	8.1%
Verizon	78	7.8%
Branch Metrics	70	7.0%
Amazon	44	4.4%
Liftoff	44	4.4%

Notably, all of the top data recipients are either in the advertising business themselves or offer data collection/processing services primarily for the purpose of advertising or otherwise monetizing users' user information. Initially, we assumed that Amazon, which does offer advertising services,[49] is primarily on this list due to its cloud hosting services (e.g., traffic to `*.amazonaws.com` or `*.elasticbeanstalk.com` likely represents *first-party* data collection). However, upon looking at the number of apps transmitting data to various Amazon-owned hostnames, it was clear that this is not the case—Amazon is on this list of top data recipients during the testing period due to its advertising services:

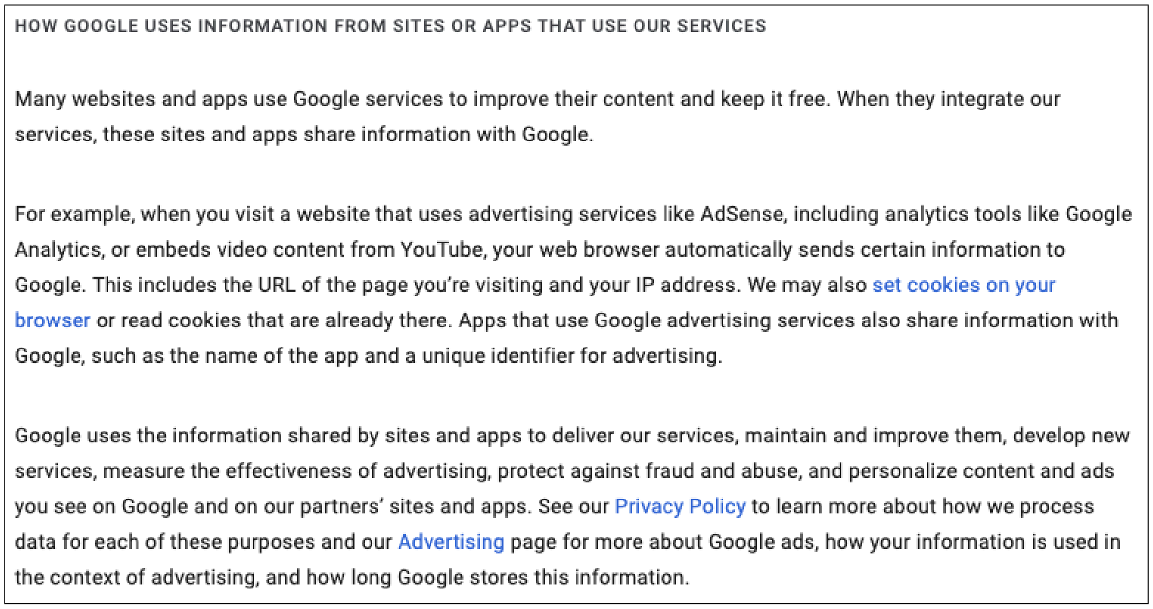
- s.amazon-adsystem.com (37 apps)
- aax-us-east.amazon-adsystem.com (32 apps)
- mads.amazon-adsystem.com (11 apps)
- aax.amazon-adsystem.com (3 apps)
- aax-us.amazon-adsystem.com (2 apps)

[48]In Figures 12 and 13, the top recipients are shown based on the average of the percentages of apps in each category that were observed sending the recipients user information during testing. In contrast, Table 3 shows the top recipients based on the total number of apps, regardless of category. Thus, this is the reason why the sets of top recipients differ (i.e., the former is averaged by category, whereas the latter is in the aggregate).

[49]`https://advertising.amazon.com/`

Google, one of the top 10 data recipients, like Amazon, offers both cloud computing and advertising services. In the list of hostnames that we observed receiving user information, the primary receiving hostname was `settings.crashlytics.com`, which suggests that this data is being used for debugging analysis:

- settings.crashlytics.com (58 apps)
- reports.crashlytics.com (8 apps)
- www.googleadservices.com (8 apps)
- googleads.g.doubleclick.net (8 apps)
- firebaseremoteconfig.googleapis.com (7 apps)



HOW GOOGLE USES INFORMATION FROM SITES OR APPS THAT USE OUR SERVICES

Many websites and apps use Google services to improve their content and keep it free. When they integrate our services, these sites and apps share information with Google.

For example, when you visit a website that uses advertising services like AdSense, including analytics tools like Google Analytics, or embeds video content from YouTube, your web browser automatically sends certain information to Google. This includes the URL of the page you're visiting and your IP address. We may also set cookies on your browser or read cookies that are already there. Apps that use Google advertising services also share information with Google, such as the name of the app and a unique identifier for advertising.

Google uses the information shared by sites and apps to deliver our services, maintain and improve them, develop new services, measure the effectiveness of advertising, protect against fraud and abuse, and personalize content and ads you see on Google and on our partners' sites and apps. See our Privacy Policy to learn more about how we process data for each of these purposes and our Advertising page for more about Google ads, how your information is used in the context of advertising, and how long Google stores this information.

Figure 14: Google disclosure indicating that data collected from Google services used by mobile apps will be used for advertising purposes.[50]

Google's disclosures indicate that when using a mobile app containing Crashlytics, Google processes the received data on behalf of app developers and does not use it for other purposes, unless it is used in conjunction with other Google services.[51] This policy is different for different Google products: disclosures for both Google Analytics for Firebase[52] and Google Analytics[53] instruct developers to display "a prominent link to the site

[50] *Ibid.*

[51] `https://web.archive.org/web/20200714154913/https://firebase.google.com/terms/crashlytics-app-distribution-data-processing-terms/`

[52] `https://web.archive.org/web/20200711041548/https://firebase.google.com/terms/analytics`

[53] `https://web.archive.org/web/20200716140718/https://marketingplatform.google.com/about/analytics/terms/us/`

`How Google uses data when you use our partners' sites or apps."'[54] This linked website discloses that when users interact with websites and/or mobile apps containing Google's services (e.g., Google Analytics), the information that is transmitted to Google will be used for advertising purposes (Figure 14).

We note that we observed few apps transmitting known user information to Google's advertising hostnames (e.g., `www.googleadservices.com` and `googleads.g.doubleclick.net`), that we were able to detect. Examining the full traffic logs, we detected over 200 apps that made *connections* to `googleads.g.doubleclick.net`: over 30% of apps in the Health category (Appendix B.1.1), 15% of Kids apps (Appendix C.1.1), and almost 27% of all Other apps (Appendix D). Thus, we suspect that our observations of clear examples of user information flowing to Google may be undercounts; based on the limitations that we discuss in Section 5, we suspect that either persistent identifiers are being obfuscated in a manner that we have yet to detect or data is being transmitted through unmonitored encrypted channels (e.g., QUIC).

[54] `https://web.archive.org/web/20200716103239/https://policies.google.com/technologies/partner-sites`

4.2.1 Privacy-Related Configuration Options in Traffic

As we noted in Section 4.1.3, many SDKs that process user information offer developers privacy-related configuration options. This way, developers can choose to disable certain behaviors entirely, or at least delay them until after the user has provided informed consent. In that previous section, we examined the usage of these options in apps' code based on static analysis (i.e., scanning the SDK's code for references to these functions). In this section, we examine the presence of privacy-related configuration options in apps' traffic that we captured during testing.

Facebook was the most prevalent recipient of user information from the apps that we tested. In their documentation, they offer app developers the ability to either disable automatic tracking of users entirely, or to disable it temporarily, so that the app may only begin tracking after the user has consented. According to their API documentation,[55] if the Facebook SDK has been correctly configured to disable tracking by default, it will transmit a parameter to Facebook's servers, "`application_tracking_enabled`," set to `false`. Examining the traffic that was transmitted to `graph.facebook.com` containing identified user information, we observed only 4 instances (1.2% of all apps sending this configuration option) where developers signaled to Facebook that tracking should be disabled by default:

1. "Stan" (au.com.stan.and; 1,000,000 installs), developed by Stan Entertainment Pty Ltd
2. "Life360 - Family Locator, GPS Tracker" (com.life360.android.safetymapd; 50,000,000 installs), developed by Life360
3. "Opera browser with free VPN" (com.opera.browser; 100,000,000 installs), developed by Opera
4. "Opera Mini - fast web browser" (com.opera.mini.native; 100,000,000 installs), developed by Opera

All of these apps were in the Other category. Separately, we identified another 312 apps that transmitted this same option set to `true` (i.e., automatically signaling to Facebook, as soon as the app is run, that the user's personal information can be used for tracking and profiling purposes). These apps were split across all three categories: 33 Health apps, 11 Kids apps, and 268 Other apps. So while Kids apps were less likely to enable tracking by default during the testing period (i.e., a third less likely than Health apps, and twenty-four times less likely than Other apps), more than 10% of Kids apps were signaling to Facebook that their users should be profiled and tracked.

[55]`https://web.archive.org/web/20190902212713/https://developers.facebook.com/docs/graph-api/reference/application/activities/`

Example

AppsFlyer Attribution Tracking

AppsFlyer was another top recipient of user information during testing. They perform services for the advertising industry, primarily "attribution marketing," and "anti ad-fraud" where they track user behavior to determine whether or not an ad campaign was successful. For example, when an ad appears in an app, the user's AAID may get sent to AppsFlyer's servers along with information indicating that a particular ad was shown to this user. If this user then clicks the ad, AppsFlyer is notified again, with a message indicating that this particular user has now clicked the ad. If this ad was for another app, and the user subsequently installs that app that was advertised in the ad, it is likely that this app, too, contains the AppsFlyer SDK. When that newly-installed app is first run, AppsFlyer will again receive the user's AAID, along with a message to indicate that this user has now installed this other app. At that point, AppsFlyer will conclude that the original ad was effective (i.e., it resulted in the user installing the advertised app), and that someone should get paid. All of this cross-app tracking—AppsFlyer also advertises the ability to do this type of tracking across devices[a]—is made possible by the fact that they are able to link a user's activities using the AAID (or another persistent identifier).

The AppsFlyer SDK transmits a parameter to its servers labeled "`advertiserIdEnabled`," which is set to `false` when the app wants to signal that the AAID should not be used for tracking or profiling purposes (e.g., because the user has enabled the system-wide opt-out, which we introduced in Section 1). We scanned all of the traffic from apps going to AppsFlyer's servers and observed no instances where this parameter was set to `false`, but 107 instances where it was set to `true`.

[a] https://web.archive.org/web/20200619152250/https://www.appsflyer.com/product/people-based-attribution

4.3 Appropriate Use of the Android Permissions System

Many of the features provided to developers by the Android platform (e.g., the Android "application programming interfaces" or "APIs")[56] are regulated by the Android permissions system because they give apps access to user information, device data, and hardware features that are potentially sensitive. This system gives users a choice (and, therefore, responsibility) to manage access to various features by accepting or rejecting corresponding permissions as they are requested by apps. In particular, Android labels several permissions as "dangerous"[57] when they have the potential to be invasive,[58] and the user must select "allow" on a standard permission-request popup to enable their use by each app when a feature that requires them is about to be used. In order to be able to request a permission, developers must declare the permission in a place common to all Android apps: the `AndroidManifest.xml` file. Thus, by reading this file, one can determine the upper bound for the permissions that an app *may* request at runtime.

It is important to note that declaration and use of "dangerous" permissions (neither the specific types, nor the quantity) does not necessarily indicate privacy-invasive behavior. Many apps available on the Google Play Store are designed to take full advantage of the Android system and its vendor-specific variants, and therefore request the use of tens or hundreds of permissions. Other apps make heavy use of a handful of extremely sensitive permissions in order to enable common features, like taking profile pictures, navigating a forest hike, or recording a voice memo. The use of these permissions is not inherently dangerous if app developers take steps to maintain and protect their usage and provide transparency to users about when they will be used and how. Developers are also responsible for ensuring that access to sensitive data, enabled by granted permissions, is not abused by the third-party SDKs that they also choose to integrate with their apps.

However, it is equally important to note that once an Android app has been granted a permission by a user, it does not need to ask for permission again (unless the user intervenes by explicitly revoking the permission), even if the data is used for other purposes. When an app receives an update, previously-granted permissions do not need to be granted again, even if the update results in the data being used in new and inappropriate ways. As such, the user (and perhaps the app developer, in the case of bundled third-party SDKs) cannot be certain that previously-acquired permissions and the sensitive data and features they govern will not be abused by embedded third-party libraries or code that is downloaded at runtime, nor that the privacy-sensitive features and data collection originally requested by the app developer remain consistent. Similarly, app and SDK developers may take advantage of impatient or unknowledgeable users seeking to gain access to features through permissions that do not actually support core app functionality. Our analysis attempts to paint a picture of permission use across the app ecosystem and discusses specific examples that may overstep the boundaries of acceptable permission use.

[56] https://en.wikipedia.org/wiki/Application_programming_interface

[57] Throughout this report, when we refer to "dangerous permissions," we are referring to those on this list.

[58] https://web.archive.org/web/20200712090715/https://developer.android.com/guide/topics/permissions/overview

Background Information

What are the "dangerous" permissions?[a]

Permission Name	Description
ACCEPT_HANDOVER	Allows a calling app to continue a call which was started in another app. An example is a video calling app that wants to continue a voice call on the user's mobile network.
ACCESS_BACKGROUND_LOCATION	Allows an app to access location in the background. If you're requesting this permission, you must also request either `ACCESS_COARSE_LOCATION` or `ACCESS_FINE_LOCATION`. Requesting this permission by itself doesn't give you location access.
ACCESS_COARSE_LOCATION	Allows an app to access approximate location.
ACCESS_FINE_LOCATION	Allows an app to access precise location.
ACCESS_MEDIA_LOCATION	Allows an application to access any geographic locations persisted in the user's shared collection.
ACTIVITY_RECOGNITION	Allows an application to recognize physical activity.
ADD_VOICEMAIL	Allows an application to add voicemails into the system.
ANSWER_PHONE_CALLS	Allows the app to answer an incoming phone call.
BODY_SENSORS	Allows an application to access data from sensors that the user uses to measure what is happening inside his/her body, such as heart rate.
CALL_PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call.
CAMERA	Required to be able to access the camera device.
GET_ACCOUNTS	Allows access to the list of accounts in the Accounts Service.
PROCESS_OUTGOING_CALLS	Allows an application to see the number being dialed during an outgoing call with the option to redirect the call to a different number or abort the call altogether.
READ_CALENDAR	Allows an application to read the user's calendar data.
READ_CALL_LOG	Allows an application to read the user's call log.
READ_CONTACTS	Allows an application to read the user's contacts data.
READ_EXTERNAL_STORAGE	Allows an application to read from external storage.
READ_PHONE_NUMBERS	Allows read access to the device's phone number(s). This is a subset of the capabilities granted by `READ_PHONE_STATE` but is exposed to instant applications.
READ_PHONE_STATE	Allows read only access to phone state, including the phone number of the device, current cellular network information, the status of any ongoing calls, and a list of any PhoneAccounts registered on the device.
READ_SMS	Allows an application to read SMS messages.
RECEIVE_MMS	Allows an application to monitor incoming MMS messages.
RECEIVE_SMS	Allows an application to receive SMS messages.
RECEIVE_WAP_PUSH	Allows an application to receive WAP push messages.
RECORD_AUDIO	Allows an application to record audio.
SEND_SMS	Allows an application to send SMS messages.
USE_SIP	Allows an application to use SIP service.
WRITE_CALENDAR	Allows an application to write the user's calendar data.
WRITE_CALL_LOG	Allows an application to write (but not read) the user's call log data.
WRITE_CONTACTS	Allows an application to write the user's contacts data.
WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage.

[a] `https://web.archive.org/web/20200712090715/https://developer.android.com/guide/topics/permissions/overview`

4.3.1 Permission Usage

Our testing environment records which permissions were declared by each app, and logs usage of many common dangerous permissions. In this section, we compare the dangerous permissions that apps in each category requested and used during testing, which are depicted in Figures 15 and 16, respectively.

In Appendix F, for each app, we provide data on all permissions requested and those detected as being used during our testing. Appendices B.2, C.2, and D list the top permissions requested by apps in the Health, Kids, and Other categories. Overall, across all apps tested, READ_CONTACTS, READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE were the most commonly requested and used permissions. While the first governs access to the user's address book contacts, the latter two govern access to the phone's shared internal storage, which is an area of the phone's filesystem that allows apps to read and write files that are accessible to other apps—provided that they, too, possess the requisite permissions. This ability has many legitimate uses. For example, because the phone's photo library resides on the shared filesystem, any app that requires access to photos taken via the camera app will need to be granted the READ_EXTERNAL_STORAGE permission. On the other hand, this permission can also be abused, which we discuss in Section 4.4. Across all of the app categories, a majority of apps were observed using these permissions, with the exception of READ_EXTERNAL_STORAGE among Kids apps.

Another prevalent permission observed being used during the testing period was ACCESS_FINE_LOCATION, which governs access to the phone's GPS sensor. Almost half of apps in the Health and Other categories (54% and 41%, respectively), accessed the type of location data that would allow them to share GPS Location. That we noticed precipitously fewer—less than a tenth—transmissions of GPS Location in Section 4.1, may be an indication that either most of this data is being used internally by the app (i.e., and simply not transmitted off of the device), its transmission is being obfuscated in a manner undetectable by us, or that the granularity of the data is being coarsened prior to transmission. That is, if an app only needs to share an *approximate* location with an Internet server, our system will not identify it as GPS Location if it is more than 100m from the device's physical location.

Table 5 depicts the apps that requested the most permissions. As can be seen, these app were almost entirely "productivity"[59] apps found in the Other category, such as various launchers and messaging tools.

[59]Google Play's app categorization is defined by the app developer at the time of publishing its software through the Google Play Developer Console.

Table 5: Apps that requested the most permissions during testing.

App Information	Requested Permissions	Dangerous Permissions	Manual Categorization	Google Play Category
"Parallel Space - Multiple accounts & Two face" (com.lbe.parallel.intl; 100,000,000 installs), developed by LBE Tech	102	21	Launcher	Personalization
"SureLock Kiosk Lockdown" (com.gears42.surelock; 100,000 installs), developed by 42Gears Mobility Systems	99	11	Launcher	Business
"DO Multiple Accounts - Infinite Parallel Clone App" (do.multiple.cloner; 1,000,000 installs), developed by River Stone Tech	90	14	Launcher	Tools
"Multi Parallel - Multiple Accounts & App Clone" (multi.parallel.dualspace.cloner; 1,000,000 installs), developed by Winterfell Applab - Clone App & Status Downloader	86	14	Launcher	Tools
"Samsung Smart Switch Mobile" (com.sec.android.easyMover; 100,000,000 installs), developed by Samsung Electronics Co., Ltd.	65	13	Utility	Tools
"POCO Launcher 2.0 - Customize, Fresh & Clean" (com.mi.android.globallauncher; 10,000,000 installs), developed by Xiaomi Inc.	57	8	Launcher	Personalization
"Google" (com.google.android.googlequicksearchbox; 5,000,000,000 installs), developed by Google LLC	57	15	Utility	Tools
"TextU - Private SMS Messenger, Call app" (com.textu.sms.privacy.messenger; 100,000 installs), developed by Melons Chat Group	54	20	Messenger	Communication
"Mobile Security: VPN Proxy & Anti Theft Safe WiFi" (com.wsandroid.suite; 50,000,000 installs), developed by McAfee LLC	53	12	Privacy & Security	Productivity

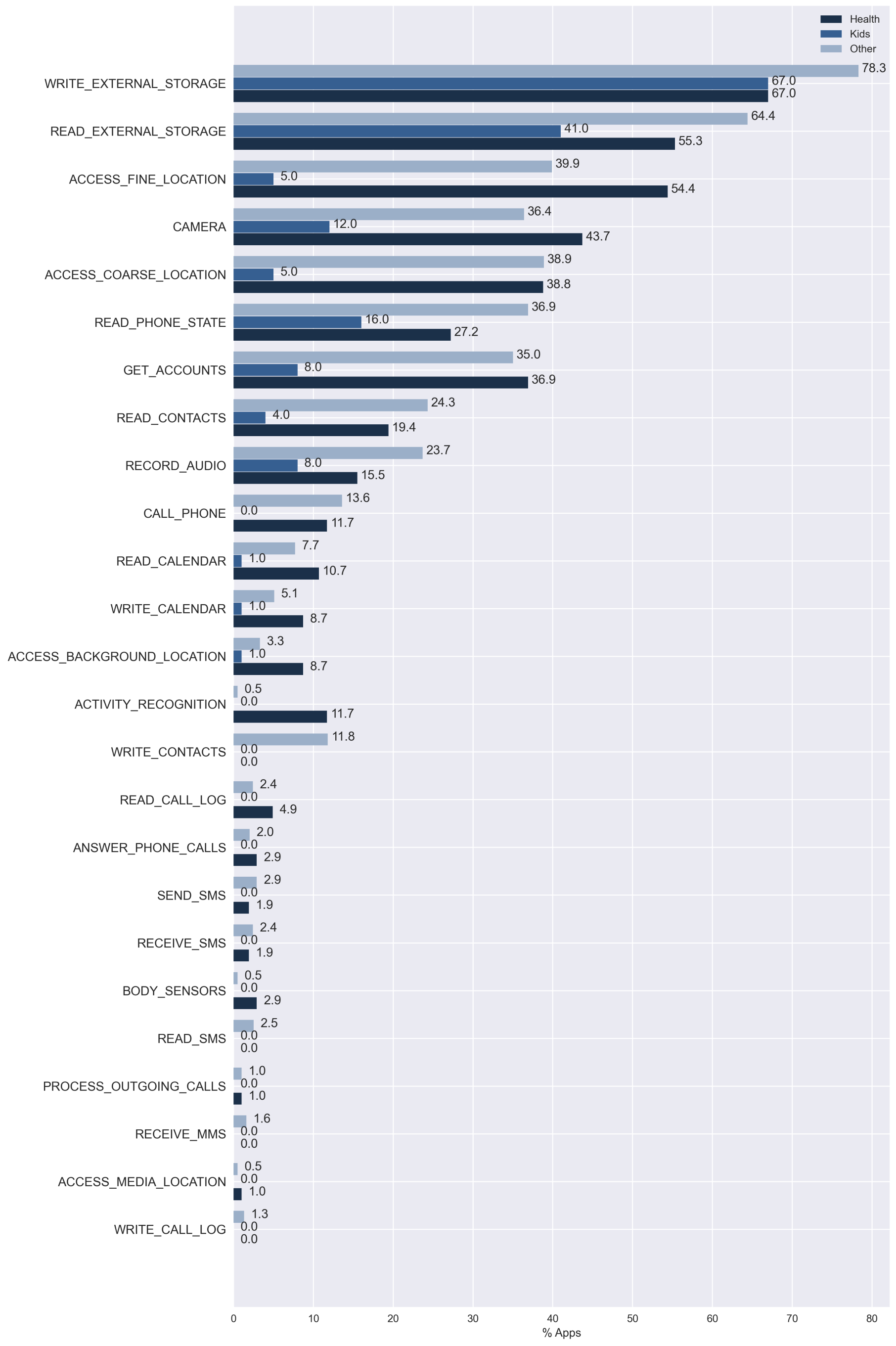


Figure 15: Percentage of apps *requesting* dangerous permissions during testing.

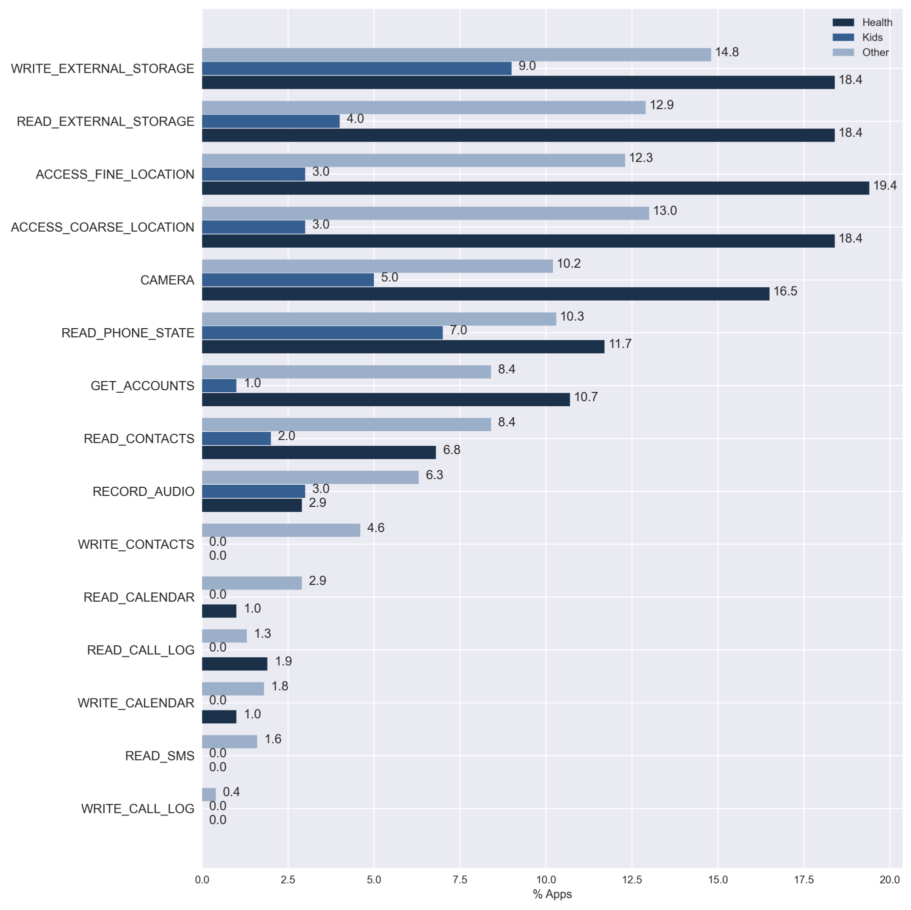


Figure 16: Percentage of apps *using* dangerous permissions during testing.

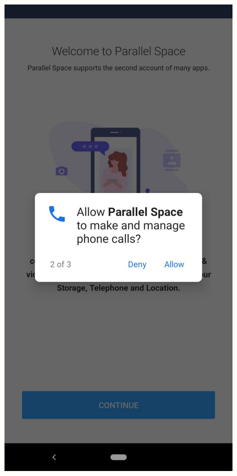
Example

Permissions and Expected Functionality: Parallel Space

It may be surprising that some apps were observed requesting so many permissions. For example, "Parallel Space - Multiple accounts & Two face" (com.lbe.parallel.intl; 100,000,000 installs), developed by LBE Tech, was observed requesting 102 permissions, spanning the gamut of what is possible on the Android system. Of those permissions, 21 were dangerous[a] (≈ 70% of the dangerous permissions we are able to track), including:

ACCEPT_HANDOVER	BODY_SENSORS	READ_PHONE_STATE
ACCESS_BACKGROUND_LOCATION	CALL_PHONE	RECORD_AUDIO
ACCESS_COARSE_LOCATION	CAMERA	USE_SIP
ACCESS_FINE_LOCATION	GET_ACCOUNTS	WRITE_CALENDAR
ACCESS_MEDIA_LOCATION	READ_CALENDAR	WRITE_CONTACTS
ACTIVITY_RECOGNITION	READ_CONTACTS	READ_EXTERNAL_STORAGE
ANSWER_PHONE_CALLS	READ_PHONE_NUMBERS	WRITE_EXTERNAL_STORAGE

In context, this amount of permission usage may very well be appropriate, since the features and functionality the app provides to its users is complex and varied. In their Google Play description, they write:


Welcome to Parallel Space
Parallel Space supports the second account of many apps.
Allow Parallel Space to make and manage phone calls?
2 of 3
Deny
Allow
Storage, Telephone and Location.
CONTINUE

> "Parallel Space needs to apply for the permissions required by the apps added in Parallel Space to function normally. For example, if Parallel Space is not permitted to acquire your location, you will be unable to send your location to your friends in some apps that run in Parallel Space. Parallel Space does not collect your personal information to protect privacy."[b]

It should also be noted that our testing environment captured transmissions of the AAID from Parallel Space to third parties like AppsFlyer and Facebook, and the WiFi MAC and AAID to LBE, the app publisher. Normally, no dangerous permissions are required to access these identifiers, and so their collection is relatively invisible to the user.

[a]See Section 4.3 for an explanation of "dangerous permissions."

[b]`https://web.archive.org/web/20200712144458/https://play.google.com/store/apps/details?id=com.lbe.parallel.intl`

Example

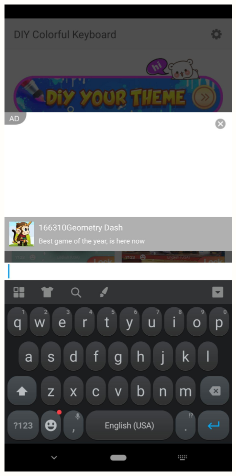
Permissions and Expected Functionality: DIY Color Keyboard

Many of the apps that were observed requesting the most permissions were similar to the Parallel Space app, in that they are complex utilities for which many permissions may be necessary to function, such as security and privacy tools, and high-fidelity interactive communication apps. However, one particular app, "DIY color keyboard" (com.diy.keyboard; 1,000,000 installs), developed by Paredes BLACK-GOAT, requested 47 permissions, 8 of which were considered "dangerous," including:

ACCESS_COARSE_LOCATION	ACCESS_FINE_LOCATION	CAMERA
GET_ACCOUNTS	READ_CONTACTS	READ_EXTERNAL_STORAGE
READ_PHONE_STATE	WRITE_EXTERNAL_STORAGE	

While it is conceivable that these permissions might find their way into a keyboard app to support a high degree of flexibility and customization, the other non-dangerous permissions this app had access to were also interesting, such as `WRITE_APN_SETTINGS`, `READ_LOGS`, and `MOUNT_UNMOUNT_FILESYSTEMS`.

And, again, this app was observed also sending the user's AAID to AppsFlyer, which requires no permissions and is likely invisible to the user. More interestingly, DIY Colorful Keyboard's privacy notice states:



> "Collection: You can use the Service without providing any user information. We may collect and store user information you provide to us through the Service, such as keywords entered for searches. We may also collect any communications between you and DIY Colorful Keyboard Team.
>
> "Aim: The purpose of collection of keywords is to analyse your needs and to conduct profiling, in order to make accurate advertisement placement."[a]

[a] https://web.archive.org/web/20200714225517/http://s3-us-west-2.amazonaws.com/public-policy/keyboard-color/DIY+Colorful+Keyboard+Privacy+Policy.htm

4.4 Abuses of the Android Permission System and Policies

Background Information

What are Android permissions and how can they be circumvented?

Android permissions are a way that the Android platform gives users control over how apps can access sensitive data (e.g., several types of user information presented in Section 3), protected hardware (e.g., the camera and microphone), and other functionality on their mobile devices. Permissions cover a wide range of functionality: access to contacts, photos, the camera, location data, and Bluetooth functions are some examples of the types of functionality that are controlled by permissions. Permissions provide control by giving users the ability to allow or deny an app's use of specific functionality (e.g., allowing access to precise location data, denying access to address book contacts, etc.). As explained in Section 4.3, the Android platform designates a subset of permissions as "dangerous," and requires that users be prompted for approval the first time that an app attempts to access a resource protected by a dangerous permission.

For example, if an app developer wants to build an app that puts filters on photos, the app developer would need to request access to the camera, in order to use the camera to take new pictures. The app developer would also need to request access to the device's shared storage, in order to access existing photos the user had previously taken. To request each of these permissions, the app developer must specify in their app the full list of permissions that the app *may* request at runtime. When the app is run and a dangerous permission is needed for the first time, the Android platform will prompt the user if they would like to grant the permission or not. If the permission is granted, the app gains the ability to request the protected data via the appropriate Android APIs. If the permission is denied, the app is prevented from accessing that data using Android APIs.

The Android permission system prevents apps from accessing protected data, though only to the extent that they attempt to do so using the official Android APIs. If the same data is accessible to apps via sources that are not protected by the permission system, then permissions can be circumvented. For example, if a user is asked to grant location access to an app and declines, the app may circumvent that user's decision by performing a geoIP lookup on the device's IP address, which is data not protected by the permission system. While in this particular example the precision of the location data from the geoIP lookup is likely to be greatly inferior to that of the permission-protected GPS data, it illustrates the point that if the same or similar data is available via unprotected means, the permission system can be circumvented.

4.4.1 Permission System Circumvention

Based on our previous research experience [10], we have observed mobile apps exploit software vulnerabilities that allow them to collect user data protected by the permissions system, without actually holding the requisite permissions. During our analysis of these apps, we observed two different security vulnerabilities that were exploited by some apps, resulting in circumvention of the permissions system. Both of these methods exploit what are known as "side channels." A side channel is a way for a developer to bypass explicit protections by accessing user information via alternative means. For example, if the official Android APIs require permissions to access location data, those permissions may be circumvented if location data can be found elsewhere on the device (e.g., photo metadata, system log files, etc.). One of the side channels that we identified provided access to a persistent unique identifier that is not permitted to be accessed under any circumstances,[60] whereas the other allowed access to location information without requiring the location permission.

As persistent identifiers can be used for tracking, Android restricts access to many of them. Nevertheless, some apps were observed accessing these identifiers, effectively bypassing privacy protections and the user consent process. One such identifier, the WiFi MAC, is used for network communication. This is an extremely persistent identifier that is uniquely tied to a particular phone. Users cannot change it without taking extraordinary steps: "jailbreaking" the phone and installing a custom operating system. Consequently, access to the WiFi MAC is prohibited by the Android operating system (both by policy and enforced through technical means),[61] due to these privacy implications. Nevertheless, we detected the collection and transmission of the WiFi MAC amongst several apps.

Table 6 shows a list of apps that sent the WiFi MAC during testing, and the domain to which they were observed sending it. Note that when transmissions share a destination it is typically because the same third-party library is embedded in different apps. Here, the WiFi MAC was accessed due to a Java vulnerability, and in most cases, it was sent to Internet servers belonging to Umeng. Figure 17 provides an example transmission of the WiFi MAC to Umeng. Because this is being transmitted by a third-party SDK, it is likely that the app developers were unaware that it was even occurring. Yet, numerous app developers are responsible for their roles in transmitting sensitive user information in violation of platform policies.

[60] `https://web.archive.org/web/20200510021708/https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html`

[61] *Ibid.*

Table 6: Apps observed exploiting a side channel to collect WiFi MAC, and the hostname to which they were observed transmitting it.

App Name	Destination
"AliExpress - Smarter Shopping, Better Living" (com.alibaba.aliexpresshd; 100,000,000 installs), developed by Alibaba Mobile	acs.m.taobao.com
"Apex Launcher - Customize,Secure,and Efficient" (com.anddoes.launcher; 10,000,000 installs), developed by Android Does Team	alog.umeng.com
"Bass Booster & Equalizer - Virtualizer" (com.bass.boost.equalizer.music.pro; 100,000 installs), developed by Emily Team	plbslog.umeng.com
	ulogs.umeng.com
"Brain Out –Can you pass it?" (com.mind.quiz.brain.out; 100,000,000 installs), developed by Focus apps	alogus.umeng.com
	ouplog.umeng.com
"Cross Boss" (com.kepgames.crossboss.android; 100,000 installs), developed by KEP Games AB	tracker-api.my.com
"DO Multiple Accounts - Infinite Parallel Clone App" (do.multiple.cloner; 1,000,000 installs), developed by River Stone Tech	plbslog.umeng.com
	ulogs.umeng.com
"Electric Screen for Prank Live Wallpaper &Launcher" (mobi.infolife.ezweather.livewallpaper.g2...; 10,000,000 installs), developed by Weather Widget Theme Dev Team	alog.umeng.com
"Email App for Any Mail" (park.outlook.sign.in.client; 5,000,000 installs), developed by Craigpark Limited	tracker-api.my.com
"Fine Booster - Powerful Android Optimize Tool" (com.fine.booster.carytool; 500,000 installs), developed by For World Peace	plbslog.umeng.com
	ulogs.umeng.com
"Galaxy S10 Launcher for Samsung" (com.amber.launcher.samsung.s10.galaxy.s9...; 1,000,000 installs), developed by Weather Widget Theme Dev Team	alog.umeng.com
"Geek Cleaner - Easy & Smart Cleaner" (com.geek.cleaner.francis.cony; 100,000 installs), developed by Francis Team	plbslog.umeng.com
	ulogs.umeng.com
"Handcent Next SMS (Best texting with MMS,stickers)" (com.handcent.app.nextsms; 1,000,000 installs), developed by Handcent	mobile.measurelib.com
"HelloTalk —Chat, Speak & Learn Foreign Languages" (com.hellotalk; 5,000,000 installs), developed by HelloTalk Learn Languages App	sc.hellotalk8.com
"Jewel Blast Dragon - Match 3 Puzzle" (com.go7game.jewelclassic; 100,000 installs), developed by Go7Game	plbslog.umeng.com

(Table Continues)

Table 6 (Continued)

App Information	Destination
	ulogs.umeng.com
"Local Weather Forecast" (tools.weather.forecast; 1,000,000 installs), developed by Tools Dev	alog.umeng.com
"Messenger" (messenger.social.chat.apps; 10,000,000 installs), developed by Text Free & Free Call & SMS & MMS & Messenger Team	alog.umeng.com
"Messenger for SMS" (com.link.messages.sms; 10,000,000 installs), developed by SMS Messenger	alog.umeng.com
"Mi Fit" (com.xiaomi.hm.health; 50,000,000 installs), developed by Anhui Huami Information Technology Co.,Ltd.	apilocate.amap.com
"Mobile Zone for Child Devices" (au.com.familyzone; 100,000 installs), developed by FAMILY ZONE CYBER SAFETY LIMITED	api.familyzone.com
	dc.familyzone.tools
	vs.familyzone.com
"Multi Parallel - Multiple Accounts & App Clone" (multi.parallel.dualspace.cloner; 1,000,000 installs), developed by Winterfell Applab - Clone App & Status Downloader	plbslog.umeng.com
	ulogs.umeng.com
"One Booster - Antivirus, Booster, Phone Cleaner" (com.cleanteam.oneboost; 10,000,000 installs), developed by One Family	alog.umeng.com
"One Security - Antivirus, Cleaner, Booster" (com.cleanteam.onesecurity; 5,000,000 installs), developed by One Family	alog.umeng.com
"P Launcher 2020 new ▯" (com.or.launcher.oreo; 500,000 installs), developed by N Dev Team	ulogs.umeng.com
"Parallel Space - Multiple accounts & Two face" (com.lbe.parallel.intl; 100,000,000 installs), developed by LBE Tech	aff-policy.lbesecapi.com
	aff-report.lbesecapi.com
	api.lbesecapi.com
"Super 3D Wallpapers - Super realistic 3D effects" (com.super.wallpapers.seat; 100,000 installs), developed by Seat Wallpapers Labs	plbslog.umeng.com
	ulogs.umeng.com
"Super S9 Launcher for Galaxy S9/S8/S10 launcher" (com.s9launcher.galaxy.launcher; 10,000,000 installs), developed by Super Launcher Serie	ulogs.umeng.com
"Super Security –Antivirus, AppLock, Virus Cleaner" (com.ludashi.security; 1,000,000 installs), developed by DUALSPACE studio	plbslog.umeng.com
	ulogs.umeng.com
	msg.umengcloud.com

(Table Continues)

Table 6 (Continued)

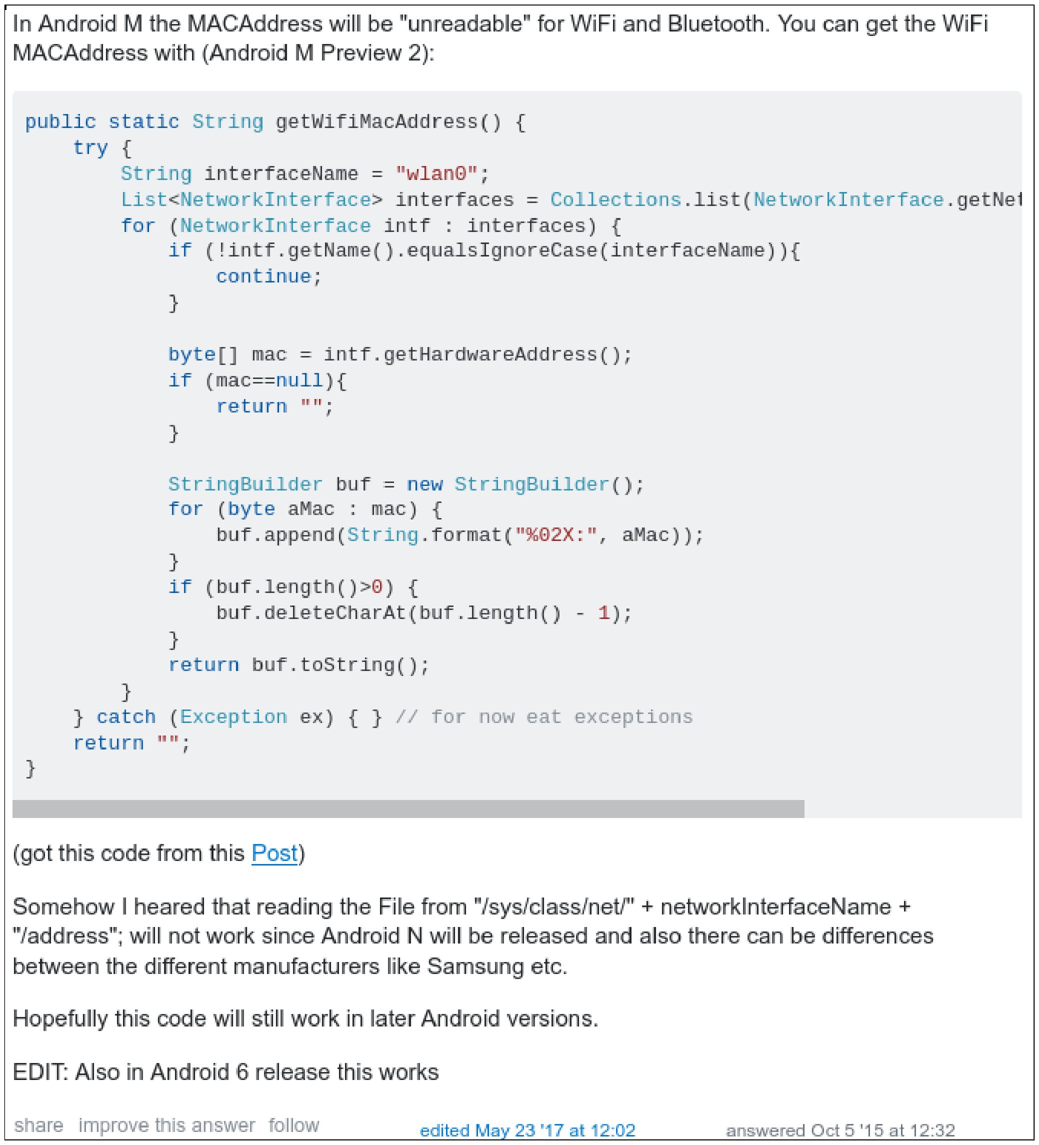
App Information	Destination
"Superb Casino - HD Free Slots Games" (com.uphill.slot.classic; 1,000,000 installs), developed by SuperB Casino	d2zntg5dbtvaei.cloudfront.net
"UC Browser- Free & Fast Video Downloader, News App" (com.UCMobile.intl; 500,000,000 installs), developed by UCWeb Singapore Pte. Ltd.	attr.union.ucweb.com
"Universal Email App" (park.hotm.email.app; 1,000,000 installs), developed by Craigpark Limited	tracker-api.my.com
"VK —live chatting & free calls" (com.vkontakte.android; 100,000,000 installs), developed by VK.com	tracker-api.my.com
"Vegas Night Slots - HOT&FREE VEGAS CASINO GAMES" (com.vegas.casino.slothbunm; 100,000 installs), developed by Zengzhineng	d1ao9yhp9fm60r.cloudfront.net
"Virus Cleaner - Antivirus, Booster & Phone Clean" (antivirus.anti.virus.cleaner.security.bo...; 5,000,000 installs), developed by Phone Clean Apps	alog.umeng.com
"Virus Cleaner-Antivirus, Phone Clean, Boost Master" (virus.cleaner.antivirus.phone.security.b...; 1,000,000 installs), developed by Super Security Studio	alog.umeng.com
"Yoho Sports" (com.uthink.ring; 10,000,000 installs), developed by mCube Inc.	plbslog.umeng.com ulogs.umeng.com
"iLauncher OS13-Phone X style" (com.amber.launcher.ios.iphonex.apple.os1...; 1,000,000 installs), developed by Weather Widget Theme Dev Team	alog.umeng.com
"myMail –Email for Hotmail, Gmail and Outlook Mail" (com.my.mail; 10,000,000 installs), developed by My.com B.V.	tracker-api.my.com
"微博" (com.sina.weibo; 10,000,000 installs), developed by Sina.com	sdkapp.mobile.sina.cn
"新浪新□" (com.sina.news; 100,000 installs), developed by Sina.com	abroad.apilocate.amap.com b.qchannel03.cn truth.qchannel03.cn beacon.sina.com.cn

```
POST /app_logs HTTP/1.1
X-Umeng-UTC: XXXXXXXXXXXXX
X-Umeng-Sdk: Android/6.1.4 Super+Antivirus/...
Msg-Type: envelope/json
Content-Type: envelope/json
User-Agent: Dalvik/2.1.0 ...
Host: alog.umeng.com
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Length: 1142

..1.0..XXXXXXXXXXXXXXXXX.@XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.............
{
  "body": {
    "activate_msg": {
      "ts": "XXXXXXXXXXXXXXX",
    }
  }
  "header": {
    "access": "wifi",
    "app_signature": "A1:1C:98:34:20:99:B6:B7:C7:CE:8C:63:69:17:F5:AA",
    "app_version": "1.3.3",
    "appkey": "5c3d81bdf1f556a763000a54",
    "carrier": "",
    "channel": "virus.cleaner.antivirus.phone.security.boost.googleplay",
    "device_id": "XXXXXXXXXXXXXXX",
    "device_manufacturer": "Google",
    "device_manuid": "XXXXXXXXXXXX",
    "device_manutime": "1583428213000",
    "display_name": "Super Antivirus",
    "id_tracking": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX==\{}n",
    "idmd5": "XXXXXXXXXXXXXXXXXXXXXXXX",
    "language": "en",
    "mc": "XX:XX:XX:XX:XX:XX",
    "os_version": "9",
    "package_name": "virus.cleaner.antivirus.phone.security.boost",
    "req_time": "0",
    "timezone": "-8",
    "version_code": "133",
  }
}
```

Figure 17: Example transmission from Virus Cleaner-Antivirus, Phone Clean, Boost Master app to Alibaba-owned Beijing Ruixin Lingtong Technology Co's `alog.umeng.com` domain. We have added whitespace and redactions. The device's WiFi MAC appears in red.

This side channel has been known since 2015,[62] and a description of how to exploit it appears on the programming advice website "Stack Overflow," with a person asking how to obtain accurate WiFi MAC addresses on newer versions of Android. Figure 18 presents a reply to this question, dated 2015.



In Android M the MACAddress will be "unreadable" for WiFi and Bluetooth. You can get the WiFi MACAddress with (Android M Preview 2):

```
public static String getWifiMacAddress() {
    try {
        String interfaceName = "wlan0";
        List<NetworkInterface> interfaces = Collections.list(NetworkInterface.getNet
        for (NetworkInterface intf : interfaces) {
            if (!intf.getName().equalsIgnoreCase(interfaceName)){
                continue;
            }

            byte[] mac = intf.getHardwareAddress();
            if (mac==null){
                return "";
            }

            StringBuilder buf = new StringBuilder();
            for (byte aMac : mac) {
                buf.append(String.format("%02X:", aMac));
            }
            if (buf.length()>0) {
                buf.deleteCharAt(buf.length() - 1);
            }
            return buf.toString();
        }
    } catch (Exception ex) { } // for now eat exceptions
    return "";
}
```

(got this code from this Post)

Somehow I heared that reading the File from "/sys/class/net/" + networkInterfaceName + "/address"; will not work since Android N will be released and also there can be differences between the different manufacturers like Samsung etc.

Hopefully this code will still work in later Android versions.

EDIT: Also in Android 6 release this works

share improve this answer follow
edited May 23 '17 at 12:02
answered Oct 5 '15 at 12:32

Figure 18: Stack Overflow article describing how to circumvent the security measures on Android to obtain the device's WiFi MAC.

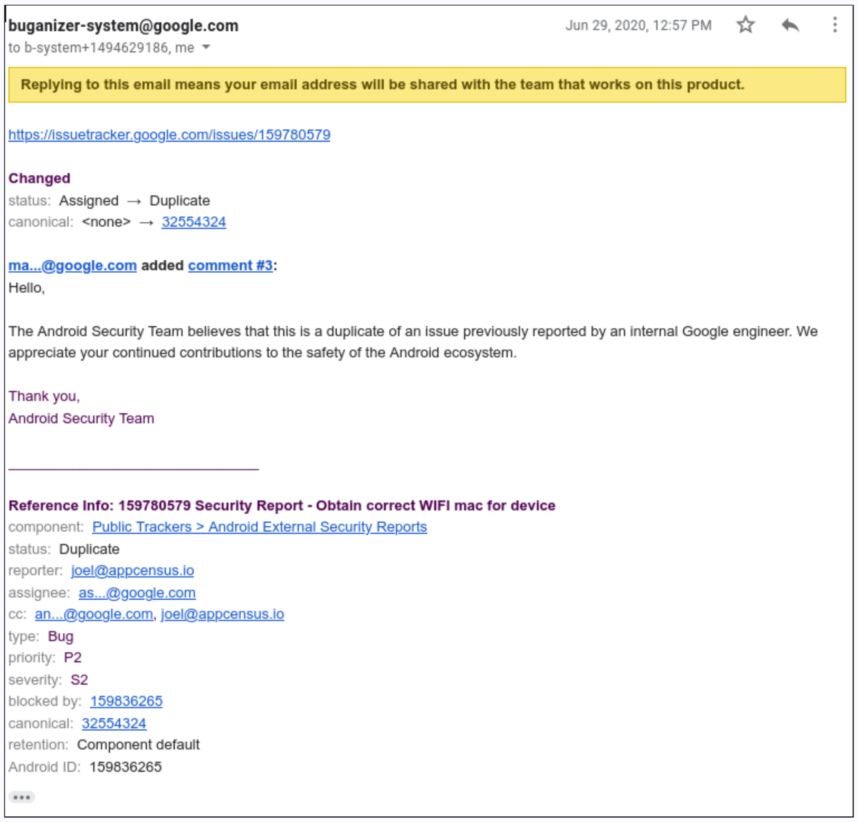
[62] https://web.archive.org/web/20200531105819/https://stackoverflow.com/questions/31329733/how-to-get-the-missing-wifi-mac-address-in-android-marshmallow-and-later

Background Information

Are current versions of Android still vulnerable to WiFi MAC collection?

We tested to see if this side channel was still exploitable on a brand new Android phone running a stock version of Android 10. We did this because Google only allows bug bounties for security issues that impact a version of Android released within the last 30 days—despite these builds only being used by a tiny fraction of Android users.[a]

We found that it was *still exploitable*, and so we filed a full bug report with their bug bounty program. Our report included the full source code for a demonstration Android app that exploited the vulnerability and detailed its impact on the Android app ecosystem. The image included here shows the reply that we received. In short, Google explained that this issue is a duplicate to *another issue* that was *already known* to them, and thus our report is ineligible for a bug bounty. The vulnerability has been exploited since 2015, and as of this writing, continues to be.



buganizer-system@google.com Jun 29, 2020, 12:57 PM
to b-system+1494629186, me

Replying to this email means your email address will be shared with the team that works on this product.

https://issuetracker.google.com/issues/159780579

Changed
status: Assigned → Duplicate
canonical: <none> → 32554324

ma...@google.com added comment #3:
Hello,

The Android Security Team believes that this is a duplicate of an issue previously reported by an internal Google engineer. We appreciate your continued contributions to the safety of the Android ecosystem.

Thank you,
Android Security Team

Reference Info: 159780579 Security Report - Obtain correct WIFI mac for device
component: Public Trackers > Android External Security Reports
status: Duplicate
reporter: joel@appcensus.io
assignee: as...@google.com
cc: an...@google.com, joel@appcensus.io
type: Bug
priority: P2
severity: S2
blocked by: 159836265
canonical: 32554324
retention: Component default
Android ID: 159836265

[a] https://web.archive.org/web/20200619141937/https://www.androidauthority.com/android-version-distribution-748439/

The second side channel we found was used to gather the WiFi router's SSID (WiFi SSID), which can be used to infer users' locations. The Android operating system protects this information by requiring that the user grant permission for the app to access location information prior to the app being allowed to access the WiFi SSID via the appropriate Android API. We observed, however, that one app transmitted this data despite not having been granted the required permission. The app, "CNN Breaking US & World News" (com.cnn.mobile.android.phone; 10,000,000 installs), developed by CNN, transmitted the WiFi SSID to `metrics.claspws.tv` (Clasp TV) and `mobile.eum-appdynamics.com` (Appdynamics). We investigated how this happened and found that a system function existed to give updates when the WiFi networks change. This function happened to return extra information about the WiFi network, including the WiFi SSID , and did not check to see whether the app should be given this information. That is, the operating system did not check to see whether the user had granted the app a location permission. Figure 19 shows the actual transmission that exposed the existence of this side channel.

```
GET /v1/event?data=eyJldmVudCI6IkRF....= HTTP/1.1
ADRUM_1: isMobile:true
ADRUM: isAjax:true
User-Agent: Dalvik/2.1.0 ...
Host: metrics.claspws.tv
Connection: Keep-Alive
Accept-Encoding: gzip
```

```
{
  "event": "DEVICE_DISCOVERY_STARTED",
  "properties": {
    "APP_ID": "vzb2017810",
    "EXTERNAL_IP_ADDRESS": "XXX.XXX.XXX.XXX",
    "GEO_LAT": "UNKNOWN",
    "GEO_LONG": "UNKNOWN",
    "IDFA": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "REMOTE_DEVICE_ID": "android:XXXXXXXXXXXXXXX",
    "REMOTE_DEVICE_MAKE": "Google",
    "REMOTE_DEVICE_MODEL": "AOSP on sargo",
    "REMOTE_DEVICE_SCREEN_TYPE": "mobile",
    "REMOTE_DEVICE_TYPE": "Android",
    "REMOTE_LIMIT_AD_TRACKING": "false",
    "REMOTE_OS_VERSION": "9",
    "REMOTE_SDK_VERSION": "5.1.0",
    "SDK_ENTRY_POINT": "UNKNOWN",
    "TIMEZONE": "GMT-07:00",
    "VZB_TIMESTAMP": "XXXXXXXXXXX",
    "WIFI_BSSID": "UNKNOWN",
    "WIFI_CONNECTED": "true",
    "WIFI_SSID": "XXXXXXXXXXXXX",
    "distinct_id": "XX(SSID)XX:XX(IPADDR)XX"
  }
}
```

Figure 19: Example transmission during testing from the CNN app to `metrics.claspws.tv`. The data, as transmitted, is above. This is encoded in base64. The decoded data is below, which is a JSON object containing the WiFi SSID (redacted and in red).

For this side channel, the issue has been resolved on Android 10, though as we discovered, this vulnerability is being exploited on the earlier versions of Android installed on over 90% of Android devices.[63]

[63] `https://web.archive.org/web/20200619141937/https://www.androidauthority.com/android-version-distribution-748439/`

4.4.2 Bridging AAID Resets

The Google Play Store provides policies for app developers' use of specific unique identifiers. One of these is the Android advertising identifier (AAID). It has the format of a Java-style UUID, such as 4e398b96-c1b7-4937-942c-1ab8d4e34b3a. It is a resettable identifier, meaning that users can go through the system settings to reset it to a new value (the mobile tracking equivalent of clearing web browser cookies to prevent tracking across websites). Google's developer guidelines[64] require that this is the only identifier that is used for behavioral advertising and other tracking/profiling purposes. According to Google's Monetization and Ads policy,[65] when accessing the AAID, apps are required to check the status of this setting and then handle the AAID accordingly.

Google's policies further restrict associating this unique identifier with other persistent identifiers. Specifically, they state with regards to the association of the AAID with personally-identifiable information or other identifiers: *"The advertising identifier must not be connected to personally-identifiable information or associated with any persistent device identifier* (*for example: SSAID, MAC address, IMEI, etc.*) *without explicit consent of the user."*[66]

During our testing, a total of 323 apps appeared to be violating that policy. That is, they were observed transmitting the AAID alongside other persistent identifiers during testing, without obvious user consent. Below is the breakdown of these apps by category:

- Health: 30
- Kids: 32
- Other: 261

This constitutes 32.3% of the apps tested, with a large majority transmitting AAID with Android ID. Table 7 lists the prominent recipients, most of which belong to the Analytics and/or Advertisement categories.

Table 7: Top recipients observed receiving the AAID along with other persistent identifiers.

Domain	Parent Company	Country	# of Apps	Received Data
`data.flurry.com`	Verizon	United States	56	Android ID AAID IMEI
`api2.branch.io`	Branch Metrics	United States	36	Android ID AAID
`settings.crashlytics.com`	Google	United States	30	Android ID AAID

(Table Continues)

[64] https://web.archive.org/web/20200528143805/https://play.google.com/about/monetization-ads/ads/
[65] https://web.archive.org/web/20200709014728/https://support.google.com/googleplay/android-developer/answer/9857753
[66] https://web.archive.org/web/20200714175124/https://support.google.com/googleplay/android-developer/answer/9904549

Table 7 (Continued)

Domain	Parent Company	Country	# of Apps	Received Data
`t.appsflyer.com`	AppsFlyer	United States	30	Android ID AAID
`cdp.cloud.unity3d.com`	Unity Technologies	United States	22	Android ID AAID
`events.appsflyer.com`	AppsFlyer	United States	17	Android ID AAID
`control.kochava.com`	Kochava	United States	16	Android ID AAID
`graph.facebook.com`	Facebook	United States	14	Android ID AAID IMEI
`api.branch.io`	Branch Metrics	United States	11	Android ID AAID
`alog.umeng.com`	Alibaba	China	11	Android ID WiFi MAC AAID Serial # IMEI

Apart from Android ID, below are some of the other persistent identifiers that we observed being transmitted alongside the AAID, in decreasing order of usage:

- WiFi MAC
- IMEI
- Serial #

In our testing, we observed that a few apps had sent the AAID alongside the IMEI (and Android ID) to Facebook's `graph.facebook.com`. One such app was "Privacy Messenger - Private SMS messages, Call app" (com.melonsapp.privacymessenger; 1,000,000 installs), developed by Melons Messenger Inc., from the Other category, with Figure 20 showcasing a sample transmission.

Similarly, one of the apps in the Health category, "Yoho Sports" (com.uthink.ring; 10,000,000 installs), developed by mCube Inc., was observed transmitting Android ID and WiFi MAC alongside the AAID. Details of this are discussed in Section 4.5.

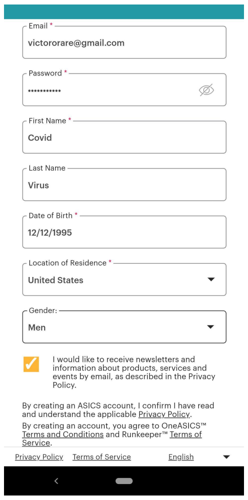
```
POST /v3.2/XXXXXXXXXXXXXX/activities HTTP/1.1
User-Agent: FBAndroidSDK.4.39.0
Accept-Language: en_AU
Content-Type: application/x-www-form-urlencoded
Content-Encoding: gzip
Transfer-Encoding: chunked
Host: graph.facebook.com
Connection: Keep-Alive
Accept-Encoding: gzip

366
format=json
&sdk=android
&custom_events=[
{
        "_eventName":"Did",
        "_eventName_md5":"74cac558072300385f7ab4dff7465e3c",
        "_logTime":XXXXXXXXXX,
        "_ui":"unknown",
        "_session_id":"XXXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
        "key":"XXXXX<IMEI>XXXX"
},
{ ... other events ...
}]
&event=CUSTOM_APP_EVENTS
&app_user_id=XXXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
&advertiser_id=XX<AAID>XX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
&advertiser_tracking_enabled=true
&anon_id=XXXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
&application_tracking_enabled=true
&extinfo=["a2","com.melonsapp.privacymessenger",624,"5.7.8","9","XXXXXXXXXX"]
&application_package_name=com.melonsapp.privacymessenger
```

Figure 20: Transmission to `graph.facebook.com` (Facebook) with both the AAID and IMEI.

4.5 In-App Privacy Disclosures

Apps notify users and acquire their consent for particular functions in a variety of ways. Our testing revealed situations where user consent was fully or mostly implied by apps, and situations where users were impeded from using the app until they explicitly consented. Figure 21 shows examples of the various ways that some of the apps we tested presented users with privacy-related disclosures.

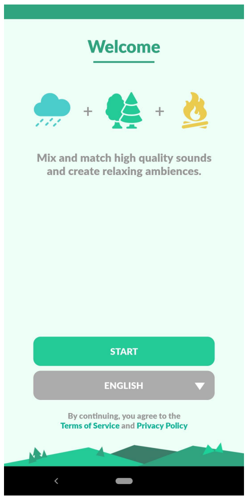


Email * victororare@gmail.com
Password * ••••••••••••
First Name * Covid
Last Name Virus
Date of Birth * 12/12/1995
Location of Residence * United States
Gender: Men

✓ I would like to receive newsletters and information about products, services and events by email, as described in the Privacy Policy.

By creating an ASICS account, I confirm I have read and understand the applicable Privacy Policy.
By creating an account, you agree to OneASICS™ Terms and Conditions and Runkeeper™ Terms of Service.

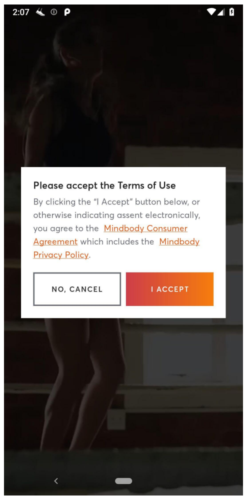
Runkeeper



Mix and match high quality sounds and create relaxing ambiences.

By continuing, you agree to the Terms of Service and Privacy Policy

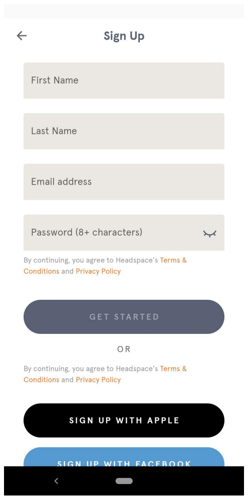
Sleepa



Please accept the Terms of Use

By clicking the "I Accept" button below, or otherwise indicating assent electronically, you agree to the Mindbody Consumer Agreement which includes the Mindbody Privacy Policy.

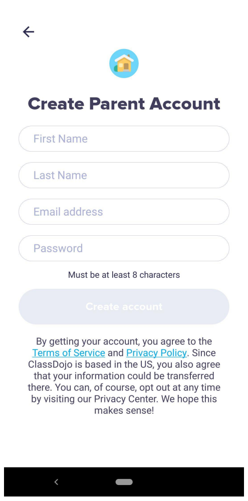
Mindbody



By continuing, you agree to Headspace's Terms & Conditions and Privacy Policy

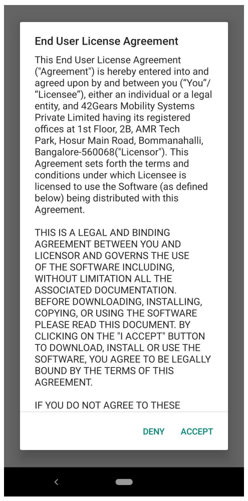
By continuing, you agree to Headspace's Terms & Conditions and Privacy Policy

Headspace



By getting your account, you agree to the Terms of Service and Privacy Policy. Since ClassDojo is based in the US, you also agree that your information could be transferred there. You can, of course, opt out at any time by visiting our Privacy Center. We hope this makes sense!

ClassDojo



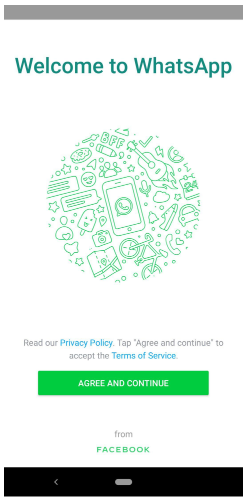
End User License Agreement

This End User License Agreement ("Agreement") is hereby entered into and agreed upon by and between you ("You"/ "Licensee"), either an individual or a legal entity, and 42Gears Mobility Systems Private Limited having its registered offices at 1st Floor, 2B, AMR Tech Park, Hosur Main Road, Bommanahalli, Bangalore-560068("Licensor"). This Agreement sets forth the terms and conditions under which Licensee is licensed to use the Software (as defined below) being distributed with this Agreement.

THIS IS A LEGAL AND BINDING AGREEMENT BETWEEN YOU AND LICENSOR AND GOVERNS THE USE OF THE SOFTWARE INCLUDING, WITHOUT LIMITATION ALL THE ASSOCIATED DOCUMENTATION. BEFORE DOWNLOADING, INSTALLING, COPYING, OR USING THE SOFTWARE PLEASE READ THIS DOCUMENT. BY CLICKING ON THE "I ACCEPT" BUTTON TO DOWNLOAD, INSTALL OR USE THE SOFTWARE, YOU AGREE TO BE LEGALLY BOUND BY THE TERMS OF THIS AGREEMENT.

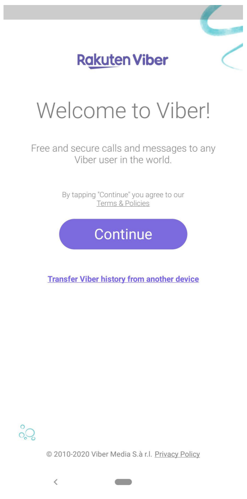
IF YOU DO NOT AGREE TO THESE

SureLock



Read our Privacy Policy. Tap "Agree and continue" to accept the Terms of Service.

WhatsApp



Free and secure calls and messages to any Viber user in the world.

By tapping "Continue" you agree to our Terms & Policies

© 2010-2020 Viber Media S.à r.l. Privacy Policy

Viber

Figure 21: Examples of privacy disclosures in various apps that we tested.

As can be seen, in most cases, to the extent that users were asked to "agree" to terms of service, the terms of service to which they were agreeing were linked in separate—often lengthy and complex—documents. Extensive research has shown us that most users are unlikely to read lengthy privacy policies (e.g., McDonald and Cranor previously showed that it would take the average Internet user multiple weeks of full-time work to read all of the privacy policies to which they are asked to agree [8]) or understand them when they do (e.g., Reidenberg et al. found that multiple experts often could not agree on how a given privacy policy should be interpreted [11]).

Due to the scale of the testing in this project, it is hard to pinpoint situations in which user consent was absolutely not acquired before collecting or transmitting user information. However, since our system logs each app's access to various types of user information, we can see how quickly apps transmitted user information after first being launched. Table 8 depicts 18 apps that transmitted the AAID within a second of the app being launched. For example, the first app transmitted the user's AAID within 100ms. Table 9 depicts the apps that transmitted other data types within 5 seconds of launch during testing.

App	Time AAID Sent (Since Launch)
"iPlayer & iWallpaper" (com.ldl.videoedit.iwallpapers; 100,000 installs), developed by Hopesalive	0:00:00.099
"Super Wallpapers Flashlight & Compass" (com.wallpaper.flashlight.compass; 500,000 installs), developed by Rio Reader LLC	0:00:00.188
"Ecosia - Trees & Privacy" (com.ecosia.android; 5,000,000 installs), developed by Ecosia.org	0:00:00.194
"Editin Photos" (com.aili.mycameras.imageedit; 1,000,000 installs), developed by Vanila iTech	0:00:00.211
"Powerful Flashlight" (com.meituan.ybw.flash; 100,000 installs), developed by SR Media Market	0:00:00.212
"My Medibank" (au.com.medibank.phs; 100,000 installs), developed by Medibank Private Limited	0:00:00.215
"Padenatef" (com.sun.newjbq.beijing.ten; 500,000 installs), developed by Holmes Shop	0:00:00.216
"Recovery all photo deleted" (com.vttl.app7.restoreimages; 1,000,000 installs), developed by VTTL	0:00:00.219
"Fashion Wallpapers" (com.fasyang.wallpaperproject; 100,000 installs), developed by Venice High	0:00:00.223
"Fine Booster - Powerful Android Optimize Tool" (com.fine.booster.carytool; 500,000 installs), developed by For World Peace	0:00:00.232
"Super Flashlight" (com.superapp.xincheng; 100,000 installs), developed by Craig's Hub	0:00:00.234
"Geek Cleaner - Easy & Smart Cleaner" (com.geek.cleaner.francis.cony; 100,000 installs), developed by Francis Team	0:00:00.243
"Pedometer" (com.baidu.news.pedometer; 100,000 installs), developed by Bright Star Lab	0:00:00.246
"Ludo Bing" (com.dotfinger.ludo.free; 100,000 installs), developed by DotFinger Games	0:00:00.247
"Brisbane Airport Guide - Flight information BNE" (com.trackingtopia.brisbaneairportguide; 100 installs), developed by TrackingTopia	0:00:00.252
"Ssafe Security - junk virus cleaner" (com.antivirus.s.security.booster.speedup; 500,000 installs), developed by Paas Dev	0:00:00.269
"Get Free Sneakers" (com.gfs21.getfreesneakers; 10,000 installs), developed by Shashel	0:00:00.279
"Apple Daily □果動新聞" (com.nextmedia; 5,000,000 installs), developed by Next Mobile Limited	0:00:00.282

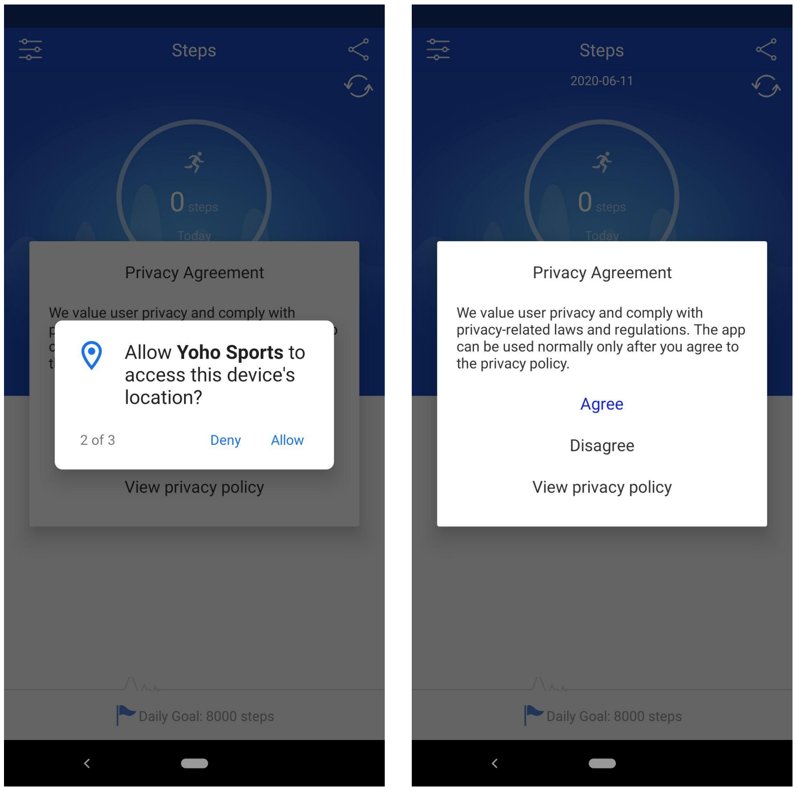
Table 8: Time since launch that apps transmitted the AAID during testing.

App	Time Sent (Since Launch)	Data Type
"PayPal Mobile Cash: Send and Request Money Fast" (com.paypal.android.p2pmobile; 100,000,000 installs), developed by PayPal Mobile	0:00:00.819	GSF ID
"Evernote - Notes Organizer & Daily Planner" (com.evernote; 100,000,000 installs), developed by Evernote Corporation	0:00:00.900	Email
"Google Drive" (com.google.android.apps.docs; 5,000,000,000 installs), developed by Google LLC	0:00:00.908	Email
"新浪新□" (com.sina.news; 100,000 installs), developed by Sina.com	0:00:01.252	WiFi MAC
"Yoho Sports" (com.uthink.ring; 10,000,000 installs), developed by mCube Inc.	0:00:01.300	WiFi MAC
"HelloTalk —Chat, Speak & Learn Foreign Languages" (com.hellotalk; 5,000,000 installs), developed by HelloTalk Learn Languages App	0:00:01.390	WiFi MAC
"Monster Numbers Full Version: Math games for kids" (com.playtic.monsternumbersFull; 10,000 installs), developed by Didactoons	0:00:01.773	BT Name
"Bass Booster & Equalizer - Virtualizer" (com.bass.boost.equalizer.music.pro; 100,000 installs), developed by Emily Team	0:00:01.837	WiFi MAC
"Overdrive Premium" (com.gsm.overdrivepremium; 1,000,000 installs), developed by GEMMOB Adventure	0:00:01.981	BT Name
"DO Multiple Accounts - Infinite Parallel Clone App" (do.multiple.cloner; 1,000,000 installs), developed by River Stone Tech	0:00:02.266	WiFi MAC
"Spider Solitaire" (com.mobilityware.spider; 10,000,000 installs), developed by MobilityWare	0:00:02.328	BT Name
"FreeCell Solitaire" (com.mobilityware.freecell; 10,000,000 installs), developed by MobilityWare	0:00:02.351	BT Name
"Multi Parallel - Multiple Accounts & App Clone" (multi.parallel.dualspace.cloner; 1,000,000 installs), developed by Winterfell Applab - Clone App & Status Downloader	0:00:02.406	WiFi MAC
"Gmail" (com.google.android.gm; 5,000,000,000 installs), developed by Google LLC	0:00:02.515	Email
"Tokaido™" (com.funforgedigital.tokaido; 100,000 installs), developed by Funforge Digital	0:00:03.096	BT Name
"Super Security –Antivirus, AppLock, Virus Cleaner" (com.ludashi.security; 1,000,000 installs), developed by DUALSPACE studio	0:00:03.192	WiFi MAC
"Social detective" (com.social.detective; 500,000 installs), developed by Profile detective tools	0:00:03.415	BT Name
"Teen Titans GO Figure!" (com.turner.ttgfigures2; 100,000 installs), developed by Cartoon Network	0:00:03.731	BT Name
"Idle Train Station Tycoon : Money Clicker Inc." (com.medu.Idle.Train.Station.Tycoon.Money...; 100,000 installs), developed by MEDU	0:00:04.353	BT Name

Table 9: Apps that transmitted user information during testing, excluding the AAID, within 5 seconds of launch, as well as the type of user information that they were observed sending.

The fitness app, Yoho Sports, is a good example of consent-seeking behavior we observed that may not match expectations. During our testing, shortly after starting the app, the user was shown a series of permission dialogs for contacts, location, and files, and then a "Privacy Agreement" dialog, which required consent to continue using the app (Figure 22). However, within 1.3 seconds of launching the app within our environment, Yoho Sports transmitted the WiFi MAC and AAID to `plbslog.umeng.com` (Alibaba). In fact, running the same app again without interacting with any permissions dialogs or privacy notices—

without touching the phone at all—yielded the same results. While the system of "notice and consent" may not be ideal (e.g., research shows that it is not [6, 16]), when users are asked to consent to a privacy policy, they would likely expect that consent would be the trigger for the data collection practices outlined in it.



Privacy Agreement

We value user privacy and comply with

Allow **Yoho Sports** to access this device's location?

2 of 3 Deny Allow

View privacy policy

Privacy Agreement

We value user privacy and comply with privacy-related laws and regulations. The app can be used normally only after you agree to the privacy policy.

Agree

Disagree

View privacy policy

Figure 22: Yoho Sports' permissions dialogs and privacy notice.

Because the phone's WiFi MAC and AAID are transmitted to the same server within the same transmission, the recipient is capable of tracking a user via their AAID, even if the user resets it (e.g., using the system setting described in Section 1). Unlike the AAID, which can be reset, the WiFi MAC persists. The collection of these two identifiers together allows Umeng (a subsidiary of Alibaba) to "bridge" various AAIDs a user has over time, and match them to the same device. Google's policies state that the AAID *"must not be connected to personally-identifiable information or associated with any persistent device identifier* (*for example: SSAID, MAC address, IMEI, etc.*) *without explicit consent of the user."*[67] Given that these two identifiers were transmitted together prior to any explicit consent occurring, it appears that this policy is being violated.

[67] https://web.archive.org/web/20200528143805/https://play.google.com/about/monetization-ads/ads/

4.6 Insecure Transmission of User Information

Transport Layer Security (TLS), formerly known as Secure Socket Layer (SSL), is an Internet standard for encrypting data as it is transmitted from one computer to another. As a standard, it has existed for over 25 years and it has been subject to multiple improvements. Its use is considered a best practice when transmitting sensitive data. Nonetheless, like with websites, not all mobile apps use TLS, and therefore sometimes transmit user information insecurely. Thus, in evaluating the data-sharing landscape, we examined whether any apps transmitted user information without encryption (i.e., without using TLS), whether this behavior was more or less prevalent among certain types of apps, what types of user information they transmitted, and to whom.

Overall, we observed that 107 apps (10.7% of all apps tested) transmitted user information without using TLS. What this means is that any computers that were involved in routing the traffic from the mobile app to the data recipient's servers (e.g., the mobile carrier, various ISPs, etc.) could potentially view any user information that was transmitted. Similarly, if the phone was connected to public WiFi, the service provider could monitor and access users' traffic. Examining the categories of apps transmitting user information insecurely, we identified 5 Health apps (4.9% of all Health apps tested), 5 Kids apps (5.0% of all Kids apps tested), and 97 Other apps (12.2% of all Other apps tested).

In terms of what that user information actually amounted to, we observed that the vast majority of insecure transmission contained the device's AAID. While insecure transmissions of the AAID are concerning, more concerning are the transmissions of identifiers that cannot easily be reset by the user. Among the apps tested, we observed that 79 apps (7.9% of all apps tested) insecurely transmitted user information other than the AAID during the testing period.

Data Recipient(s)	# Apps	Example App	Data Type(s) Transmitted
Alibaba	11	"Messenger for SMS" (com.link.messages.sms; 10,000,000 installs), developed by SMS Messenger	Android ID, WiFi MAC
Amber Weather	6	"Apex Launcher - Customize,Secure,and Efficient" (com.anddoes.launcher; 10,000,000 installs), developed by Android Does Team	Android ID, IMSI
UnderArmour	3	"Run with Map My Run" (com.mapmyrun.android2; 10,000,000 installs), developed by MapMyFitness, Inc.	Android ID
Verizon	3	"Fieldrunners 2" (com.subatomicstudios.fieldrunners2; 100,000 installs), developed by Subatomic Studios, LLC	Android ID
comScore	2	"Apple Daily □果動新聞" (com.nextmedia; 5,000,000 installs), developed by Next Mobile Limited	Android ID

Table 10: Apps observed transmitting data types (other than the AAID) insecurely.

Table 10 depicts the top 5 recipients of insecure data observed (beyond the AAID) and the number of apps transmitting data to them insecurely during the testing period. For each recipient, the table also includes examples of apps that were observed sending them data during testing (and the specific data types that we observed being sent).

As can be seen, "Messenger for SMS" (com.link.messages.sms; 10,000,000 installs), developed by SMS Messenger, transmitted the device's Android ID and WiFi MAC to servers associated with Alibaba (`alog.umeng.com`), without using TLS to secure the transmission. In terms of the app that appeared to transmit the most during testing, "新浪新□" (com.sina.news; 100,000 installs), developed by Sina.com, transmitted many different hardware-based identifiers and location data to Sina Weibo's servers, all completely unencrypted. Despite these examples of poor stewardship of user information observed during the testing period, transmitting user information without TLS remained relatively rare. These issues could be addressed in part by platforms and third-party SDKs encouraging the adoption of TLS by phasing out non-TLS API endpoints.

Background Information

What is a "cryptographic hash"?

A cryptographic hash is a "one-way" function that transposes input data of arbitrary size into a fixed size output. Given the same input data, the hash function will produce consistent output. Because of this property, it is often used to calculate "checksums" on files: the cryptographic hashes of two files can be compared to determine whether or not the two input files were identical. Hash functions are also used to obfuscate data: a known input (e.g., an IMEI) is transformed into a unique output value that no longer appears to be the IMEI (but uniquely identifies the original IMEI and shares the same degree of persistence as it relates to user tracking). Nonetheless, the cryptographic hash of a persistent identifier is yet another persistent identifier, and therefore hashing alone may not protect against identification.

In Appendix F, we detail the findings for each app that we tested, including whether or not any of its transmissions containing user information were unencrypted.

5 Limitations

While it is always our intention to collect complete data and provide accurate analysis, projects of this magnitude and complexity have inherent limitations. What follows are the challenges we faced during the course of this project, and opportunities we see for developing better techniques and processes to address them.

SDK Identification and Attribution

For various reasons, the code and assets that are packed inside an application, including third-party SDKs, are often obfuscated. The use of these techniques impedes the identification and accurate attribution of the code to their creators. Our detection methodology relies on a combination of methods that have been validated through peer-reviewed publications (e.g., [4, 7]), while also making use of public data sets (e.g., The Exodus Tracker Investigation Platform).[68] While we are constantly honing our ability to identify SDKs, our dataset is necessarily incomplete. This problem is compounded by the fact that for any given SDK, multiple versions likely exist, and different versions may have been developed by different parent companies due to mergers and acquisitions. Thus, any automated detection of third-party SDKs by its nature represents a good-faith approximation.

Also, because there is no standard of practice for building, deploying, and advertising an SDK within mobile applications, it is challenging to distill the features and purposes of SDKs into definite categories. While a particular SDK may be well-known for providing analytics functionality, it may also be used secondarily to support an advertising infrastructure, with or without the explicit declaration as such by its developer. Our efforts to classify SDKs according to the services they offer is based on the manual inspection of their websites and documentation. As a result, our analyses are generally more accurate and richer than the sources that are publicly available and the state-of-the-art in the research literature. However, we are still unable to draw a complete picture of the SDK environment. As such, we acknowledge that some SDKs may have gone undetected and others misattributed.

Endpoints and Parent Organizations

Accurately identifying the owner of an SDK or a hostname requires careful, manual inspection of sources that vary widely in fidelity. Commercial domain classification services such as McAfee's TrustedSource,[69] FortiGuard,[70] Webshrinker,[71] Dr.Web,[72] or Open DNS Domain

[68] `https://etip.exodus-privacy.eu.org/`
[69] `https://www.trustedsource.org/`
[70] `https://fortiguard.com/`
[71] `https://www.webshrinker.com/`
[72] `https://vms.drweb.com/online/`

Tagging[73] cannot be applied in this context, as they are focused on detecting security threats or potentially-harmful content, rather than classifying domains by the business of their operators. We were able to manually and confidently identify the parent company of an endpoint, its purpose, and its country-level location for roughly 82% of the hostnames contacted by the tested apps. When just identifying the primary purpose, we were able to identify 89%; whereas we were able to identify 86% of the parent companies.

In some cases, a lack of good information prevented us from being able to attribute endpoints to their operators. Hostname operators often benefit from the opacity provided by measures such as domain register anonymization or meaningless X.509 certificates (for authenticating the endpoint on TLS traffic). This is often because they belong to malicious actors, or because the organization that owns them does not want to be directly and easily associated with it (for whatever reason), or because the certificates are self-signed by the organization and not by a trusted Certificate Authority (CA). And, sometimes, identifying information is simply unnecessary. Here are some of the common problems we faced:

- No X.509 certificate was available.
- Domains can be registered via an intermediary who anonymizes its registrant.
- Domains may only be connected to companies due to legacy code.
- Domains may exist only to provide RESTful API endpoints to applications (public or otherwise), and therefore lack identifying features, whether intentionally or not.
- Many apps, services, and companies are built on the same cloud and CDN infrastructure, making it difficult to disambiguate a specific machine-generated endpoint (e.g., hostnames within `*.amazonaws.com` or `*.appspot.com`).
- Domains can be pseudo-anonymized through cloud providers, like CloudFront.

IP Geolocation

Identifying the physical locations of various API endpoints (and, by extension, the companies that govern them), is well-known to be a difficult problem. IP block markets are in constant flux, and allocations of IP blocks change (and move geographically) over time. Many IP address geolocation databases and services try to provide a consistent level of accuracy over time, but it depends on their ability to keep pace with this change, and accurately attribute addresses when changes occur. The solutions offered by commercial providers, such as MaxMind,[74] are designed to geolocate human visitors (i.e., IPs associated with ISPs offering residential or mobile access) rather than geolocating servers hosted in the cloud. For this, and many other reasons, geolocation databases have been known to differ and introduce bias.

We do not maintain our own geolocation database and technology, and must therefore rely on established ones. The accuracy of our geolocation attribution is only as good as the data we can collect from these data sources.

[73] https://community.opendns.com/domaintagging/
[74] https://www.maxmind.com/en/geoip2-databases

Permissions Used

Our testing environment uses an instrumented version of Android in which permission requests are logged and analyzed. However, the Android permission system is not fixed, and requires constant effort to understand and unravel each permission and the methods by which it is invoked. While we are always working to make our detection system as complete and accurate as possible, there may be instances where permissions that were requested and used were not logged.

Furthermore, detecting the use of permissions depends on dynamic analysis, in which apps and their features are explored and used as thoroughly and completely as possible. We employ various methods to achieve a more accurate picture of permission usage, but there may always be a feature of an app that is not easily found, or requires predetermined credentials to access. That is, dynamic analysis is a stochastic process, and therefore it is possible that under different circumstances, new functionality may be triggered that did not previously result in a certain requested permission being used. More to the point: each app was tested for a period of 10 minutes, and therefore apps that occupied that time with lengthy enrollment processes were likely to have fewer of their features examined during the remainder of the testing period.

Use of Privacy-Related Configuration Options

We measured the use of third-party SDKs' privacy-related configuration options, both in apps' source code, as well as within apps' transmissions. The generalizability of these findings is limited by the fact that we only examined the presence of these options for a handful of popular services, while also limiting our investigation to only the options described by the recipients' relevant developer documentation, and only when we were able to find it. It is possible that other configuration options may be in use that provide similar functionality, but we did not examine them because we could not find documentation of their existence and usage. For example, it is possible that some apps that we tested may be using privacy-related configuration options that we could not identify because they correspond to older SDK versions that have not been updated or because their documentation is non-public. Similarly, the use of obfuscation techniques, both in apps' code and their transmissions of user information to Internet servers, may have impeded our detection efforts. Nonetheless, amongst the privacy-related configuration options that we were able to detect, our results suggest that it is rare for app developers to configure their apps to limit the default collection of user information.

Purpose of Data Collection

Discerning whether the collection of a given data type by each mobile application is requested for a primary purpose (e.g., implementing a feature needed by the actual app) or for a secondary one (e.g., user tracking or advertising purposes) is beyond the capabilities of any mobile auditing service, including ours. In some cases, secondary purposes could be masked by primary ones. This could be the case of a hypothetical third-party SDK offering anti-fraud technologies. The access to unique identifiers could be justified to identify potentially-malicious users (e.g., users who try and register multiple accounts with a service), but could be also used for secondary purposes, like user tracking.

Our technology allows us to accurately obtain evidence of app behavior, and report that a given type of data has been uploaded from a particular device to a machine hosted in the cloud. However, once the data arrives at its destination, it is impossible for us to observe how it ultimately gets used. Answering this question without making invalid assumptions requires conducting a formal investigation into data recipients' practices. Nonetheless, inferences about potential data uses can often be made based on the recipients' posted privacy policies and marketing materials.

User Information Transmission, Encryption, and Obfuscation

While standard encryption is used in the majority of cases to transmit personal data across the Internet, there are times when apps and services employ an extra layer of obfuscation, either to provide some kind of bespoke security measure (which is generally considered bad practice), or to mask intent. In some cases, developers also use non-standard TLS libraries and protocols that prevent us from intercepting and decrypting their payloads. One case is the use of UDP-based protocols like QUIC. We identified only three applications generating traffic on UDP port 443.[75],[76],[77]

Our team works diligently to identify the various means apps and SDKs use to encrypt and obfuscate data in their traffic, but new methods are constantly being introduced.

Also, even though an SDK or API endpoint supports a type of obfuscation, that does not necessarily mean the app that transmitted data obfuscated it first. We base our overall counts of which apps appeared to use obfuscation on whether or not we found an embedded SDK or contacted an endpoint that appeared to use obfuscation.

[75] "Market+ Mobile" (com.aastocks.dzh; 1,000,000 installs), developed by AASTOCKS
[76] "Calm - Meditate, Sleep, Relax" (com.calm.android; 10,000,000 installs), developed by Calm.com, Inc.
[77] "IG Trading: Spread Betting, CFDs, Forex & Stocks" (com.iggroup.android.cfd; 500,000 installs), developed by IG Group

6 Bibliography

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

[2] Paul-Olivier Dehaye and Joel Reardon. SwissCovid: A Critical Analysis of Risk Assessment by Swiss Authorities. Technical Report arXiv:2006.10719, arXiv.org e-Print Archive, June 22 2020. https://arxiv.org/pdf/2006.10719.pdf.

[3] Erik Derr, Sven Bugiel, Sascha Fahl, Yasemin Acar, and Michael Backes. Keep me updated: An empirical study of third-party library updatability on android. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 2187–2200, New York, NY, USA, 2017. Association for Computing Machinery.

[4] Álvaro Feal, Julien Gamba, Narseo Vallina-Rodriguez, Primal Wijesekera, Joel Reardon, Serge Egelman, and Juan Tapiador. Don't accept candies from strangers: An analysis of third-party SDKs. In *Proceedings of the Computers, Privacy and Data Protection Conference* (*CPDP*), 2020.

[5] Adrienne Porter Felt, Serge Egelman, and David Wagner. I've got 99 problems, but vibration ain't one: A survey of smartphone users' concerns. In *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '12, pages 33–44, New York, NY, USA, 2012. ACM.

[6] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, New York, NY, USA, 2012. ACM.

[7] Ziang Ma, Haoyu Wang, Yao Guo, and Xiangqun Chen. Libradar: Fast and accurate detection of third-party libraries in android apps. In *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16, pages 653–656, New York, NY, USA, 2016. Association for Computing Machinery.

[8] Aleecia McDonald and Lorrie Cranor. The cost of reading privacy policies. In *Proceedings of the Technology Policy Research Conference*, September 26–28 2008.

[9] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, and Phillipa Gill. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *Proceedings of the 2018 Networking and Distributed Systems Security Symposium*, NDSS '18. Internet Society, 2018.

[10] Joel Reardon, Alvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 50 Ways to Leak Your Data: An Exploration of Apps'

Circumvention of the Android Permissions System. In *Proceedings of the 24th USENIX Security Symposium*, USENIX Security '19, Berkeley, CA, USA, 2019. USENIX Association.

[11] Joel R. Reidenberg, Travis Breaux, Lorrie Faith Cranor, Brian French, Amanda Grannis, James T. Graves, Fei Liu, Aleecia McDonald, Thomas B. Norton, Rohan Ramanath, N. Cameron Russell, Norman Sadeh, and Florian Schaub. Disagreeable privacy policies: Mismatches between meaning and users' understanding. *Berkeley Technology Law Journal*, 30(1), 2015. https://btlj.org/2015/10/disagreeable-privacy-policies/.

[12] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, and Serge Egelman. "Won't Somebody Think of the Children?" Examining COPPA Compliance at Scale. *Proceedings on Privacy Enhancing Technologies*, (2018.3):63–83, 2018.

[13] U.K. Competition & Markets Authority. Online platforms and digital advertising: Market study final report. https://www.gov.uk/cma-cases/online-platforms-and-digital-advertising-market-study, July 1 2020.

[14] Primal Wijesekera, Arjun Baokar, Ashkan Hosseini, Serge Egelman, David Wagner, and Konstantin Beznosov. Android permissions remystified: A field study on contextual integrity. In *24th USENIX Security Symposium* (*USENIX Security 15*), pages 499–514, Washington, D.C., August 2015. USENIX Association.

[15] Primal Wijesekera, Arjun Baokar, Lynn Tsai, Joel Reardon, Serge Egelman, David Wagner, and Konstantin Beznosov. The feasability of dynamically granted permissions: aligning mobile privacy with user preferences. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy*, Oakland '17. IEEE Computer Society, 2017.

[16] Primal Wijesekera, Joel Reardon, Irwin Reyes, Lynn Tsai, Jung-Wei Chen, Nathan Good, David Wagner, Konstantin Beznosov, and Serge Egelman. Contextualizing privacy decisions for better prediction (and protection). In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 1–13, New York, NY, USA, 2018. Association for Computing Machinery.

Colophon

This document was automatically generated using AppCensus' mobile app testing infrastructure. It was created using the $\LaTeX\,2_\varepsilon$ document formatting system by Leslie Lamport, based on $\TeX$ by Donald Knuth. Bibliographic references are formatted by BibTeX and the text is set in $\TeX$ Gyre Adventor.