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

Abstract: We present a scalable dynamic analysis framework that allows for the automatic evaluation of the privacy behaviors of Android apps. We use our system to analyze mobile apps' compliance with the Children's Online Privacy Protection Act (COPPA), one of the few stringent privacy laws in the U.S. Based on our automated analysis of 5,855 of the most popular free children's apps, we found that a majority are potentially in violation of COPPA, mainly due to their use of third-party SDKs. While many of these SDKs offer configuration options to respect COPPA by disabling tracking and behavioral advertising, our data suggest that a majority of apps either do not make use of these options or incorrectly propagate them across mediation SDKs. Worse, we observed that 19% of children's apps collect identifiers or other personally identifiable information (PII) via SDKs whose terms of service outright prohibit their use in child-directed apps. Finally, we show that efforts by Google to limit tracking through the use of a resettable advertising ID have had little success: of the 3,454 apps that share the resettable ID with advertisers, 66% transmit other, non-resettable, persistent identifiers as well, negating any intended privacy-preserving properties of the advertising ID.

DOI 10.1515/popets-2018-0021
Received 2017-11-30; revised 2018-03-15; accepted 2018-03-16.

***Corresponding Author: Irwin Reyes:** International Computer Science Institute, E-mail: ioreyes@icsi.berkeley.edu
Primal Wijesekera: University of British Columbia, E-mail: primal@ece.ubc.ca
Joel Reardon: University of Calgary, E-mail: joel.reardon@ucalgary.ca
Amit Elazari Bar On: University of California, Berkeley, E-mail: amit.elazari@berkeley.edu
Abbas Razaghpanah: Stony Brook University, E-mail: arazaghpanah@cs.stonybrook.edu
Narseo Vallina-Rodriguez: IMDEA Networks and International Computer Science Institute, E-mail: narseo.vallina@imdea.org
Serge Egelman: University of California, Berkeley and International Computer Science Institute, E-mail: egelman@cs.berkeley.edu

1 Introduction

In the United States, there are few comprehensive privacy regulations. However, one notable exception is the Children's Online Privacy Protection Act (COPPA), which regulates how mobile apps, games and websites are allowed to collect and process personal information from children under the age of 13 [22]. COPPA outright prohibits certain data collection practices, and requires parental consent for others. Of course, enforcement is a painstaking process, as investigations generally rely on manual examination of programs and websites to observe violations [83]. In this paper, we apply our Android dynamic analysis framework to automate the process of detecting potential COPPA violations.

Most current approaches to detecting suspicious application activity on mobile platforms rely on static analysis [*e.g.*, 33, 41, 48, 93] or dynamic analysis [*e.g.*, 28]. However, previous approaches fall short because they either do not observe actual violations, and instead only detect when a program *might* contain violative code (in the case of static analysis), or do not scale (in the case of prior dynamic analysis approaches).

We propose a new analysis framework built on prior work [67, 70, 89], which allows us to monitor *actual* program behavior in realtime and at scale. Our testing platform allows us to examine how often and under what circumstances apps and third-party libraries access sensitive resources guarded by permissions. By combining this infrastructure with a modified version of Lumen [67], an advanced network monitoring tool, we obtain a sophisticated holistic view of when sensitive data is accessed and where it gets sent.

We show that our test platform could have immediate impact on the enforcement of and compliance with COPPA (and other privacy regulations) by automating a largely manual task of identifying potential privacy violations [83]. To give an example: one observation generated from our analysis was that 37 apps—all developed by BabyBus, a company specializing in games for young children—did not access the location of the device through the standard Android permissions system. Yet, we observed them transmitting hardware and network configuration details to a Chinese analytics

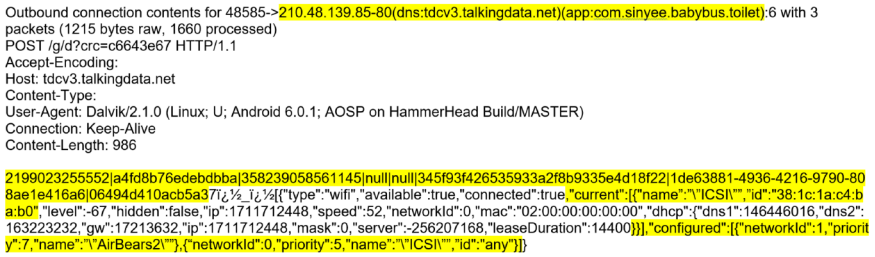

Fig. 1. TalkingData activity in BabyBus's "Toilet Training - Baby Potty": unencrypted HTTP traffic to `tdcv3.talkingdata.net`, a set of persistent identifiers, the name and MAC address of the current Wi-Fi network, and a list of configured (saved) Wi-Fi networks.

company called TalkingData (Figure 1).[1] This included the names of saved Wi-Fi hotspots and their MAC addresses, as well as the currently connected Wi-Fi access point, which can potentially be used as a surrogate for location.[2] This arguably deceptive practice is indeed well known: the United States Federal Trade Commission (FTC) reached a $4M settlement with analytics firm inMobi, for its alleged deceptive collection of location data in a very similar manner [23].

In this manner, we automatically analyzed COPPA compliance among a set of 5,855 child-directed apps from the U.S. Google Play Store. Overall, we found potential violations in a majority of these apps. Additionally, we analyzed the prevalence of potential COPPA violations in apps from companies who have been certified as COPPA-compliant under various "Safe Harbor" programs, which have come about through industry self-regulation efforts. Our analysis suggests that apps developed under Safe Harbor programs are not appreciably better, in terms of limiting potential COPPA violations.

In this paper, we describe our system in detail and then present our results from examining COPPA compliance at scale. Our contributions are as follows:

- We demonstrate a framework for fully-automated large-scale dynamic analysis of mobile apps.
- To the best of our knowledge, we perform the first large-scale analysis of COPPA compliance in the Android market using dynamic analysis.
- We show that potential COPPA violations abound: 5% of apps collect location or contact data without verifiable parental consent; 19% of apps use SDKs whose terms explicitly prohibit their use in child-directed apps (likely due to prohibited user profiling and behavioral advertising); and when SDKs do provide configuration options for COPPA compliance, those options seem to be often ignored.
- We show that industry self-regulation efforts to certify products for COPPA compliance ("Safe Harbor") do not result in more privacy-protective apps.

2 COPPA

In the United States, the 1998 Children's Online Privacy Protection Act (COPPA) governs how online services must handle data gathered from children under 13 years of age. These rules seek to protect children on the Internet through prohibitions on how online services—such as mobile apps—collect, use, and disclose personally identifiable information (PII) from children, and how parents are informed and given control over those activities. Not all online services are subject to COPPA. COPPA applies to operators of online services that are either: (i) *directed* to children under 13 years of age; (ii) directed to general audiences, but the operators have *actual knowledge* of the collection of information from users under 13; or (iii) have actual knowledge of the collection of information *directly from* users of another website or online service directed to children ("covered entities"). Operators are also subjected to COPPA for any collection or improper use performed by third-party services bundled with their core service.

One of the main objectives of COPPA is to give parents control over how their children's PII is collected and shared. COPPA has two main mechanisms to help parents make decisions about their children's data: covered entities must provide (i) a clear, understandable and complete disclosure of their PII collection practices (*i.e.*,

1 We contacted BabyBus, who responded by claiming that they had no idea this data collection was occurring, but that they have since removed the TalkingData SDK from all of their apps, which we later verified.

2 This loophole is fixed starting with Android 8.0. However, as of February 2018, Android 8.0 only accounts for 1% of the Android install base [35].

what data do they access and with whom do they share it) and (ii) direct notice and ask for *verifiable* parental consent prior to collection, usage, or disclosure of any PII, and ensure that the consenting party is in fact a legal parent or guardian. While the COPPA rule does not require one specific method to obtain consent, it does require the method be "reasonably designed in light of available technology." Disclosing personal information to third parties, such as advertising agencies, requires reliable methods of verification of parental consent, such as payment systems, signed forms, or phone calls [84].

COPPA's definition of PII is relatively broad, covering such items as contact information (*e.g.*, email addresses and phone numbers), audio or visual recordings, and precise geolocation data (*i.e.*, at the granularity of street name and city/town). Additionally, under the 2013 amendments to the COPPA rule, persistent identifiers (*e.g.*, IMEI and MAC addresses) are considered PII if they "can be used to recognize a user over time and across different websites or online services."[3]

There are certain rules that developers and third-party services must follow when using legitimately collected PII. Any PII collected from children cannot be used for profiling (*e.g.*, behavioral advertising) or cross-device tracking. However, certain limited pieces of PII may be collected without parental consent if the data is used in "support for the internal operations" of the service. The regulation defines supporting internal operations as "those activities necessary to:"[4]

> *(i) Maintain or analyze the functioning of the Web site or online service;*
> *(ii) Perform network communications;*
> *(iii) Authenticate users of, or personalize the content on, the Web site or online service;*
> *(iv) Serve contextual advertising on the Web site or online service or cap the frequency of advertising;*
> *(v) Protect the security or integrity of the user, Web site, or online service;*
> *(vi) Ensure legal or regulatory compliance; or*
> *(vii) Fulfill a request of a child as permitted by §312.5(c)(3) and (4).*

This exemption allows, for instance, third-party analytics services to gather persistent identifiers, provided that *no* other personal information is associated with it, that any identifiers collected are not used to contact or build profiles of specific users (*i.e.*, for behavioral advertising), *and* that this data collection is *necessary*.

2.1 Enforcement Actions

The FTC has moved against a number of app developers and third-party service providers for gathering PII from children: in 2014, children's app developer BabyBus received a warning about its potential collection of geolocation data [21]; in 2015, the FTC collected a $360K settlement from app studios LAI Systems, LLC and Retro Dream for allowing integrated third-party services to access and collect persistent identifiers [24]; and in 2016, the ad network InMobi was fined $1 million for gathering the locations of users—including children—without proper consent [23]. While these actions might push developers and third-party providers to be more vigilant, these are isolated incidents. Our work offers a systematic analysis of app behaviors that can help to uncover widespread misbehavior amongst apps, so that regulators and policymakers can improve accountability.

2.2 Industry Response

While COPPA places liability on operators of child-directed services, the law exempts platforms, hosting services, and distribution channels that "merely offer the public access to someone else's child-directed content."[5] Still, while the two largest app distribution platforms are therefore exempt, both the Google Play Store and the Apple App Store have implemented measures to help developers to comply with the law. Namely, developers can list their child-directed products in special child-targeted categories, provided that they observe requirements set by privacy laws and the distribution platform's terms of service. The FTC further clarifies that distribution platforms be mindful of Section 5 of the Federal Trade Commission Act, which prohibits deceptive practices, and to not "misrepresent the level of oversight they provide for a child-directed app" [22].[6]

The Google Play Store's *Designed for Families* program (DFF) is an optional review process that entitles developers to list compliant apps under those special family-friendly categories and sections specifically relevant to children under 13. Developers participating in DFF agree that "apps submitted to Designed for Families are compliant with COPPA and other relevant statutes, including any APIs that your app uses to provide the service" [34, 36]. DFF also sets restric-

3 16 C.F.R. §312.2.
4 Ibid.
5 78 Fed. Reg. 3977.
6 78 Fed. Reg. 3978.

tions on the type of content shown to children in the app (*e.g.*, avoiding sexual and violent content) and the use of in-app purchases. Our test corpus consists of apps accepted into the DFF program (and therefore the developers have actual knowledge of users under 13 in their intended audience).

COPPA also outlines a "Safe Harbor" program for industry self-regulation [81]. Industry groups are, with FTC approval, able to set and enforce privacy standards "with the same or greater protections" as COPPA, lessening the burden on industry and the FTC alike by increasing the availability and speed of the review of child-directed services. In Section 4.5, we present an in-depth analysis on the effectiveness of this program (or rather, the lack thereof).

3 Methodology

There are two main approaches to analyzing the privacy behaviors of mobile apps: static and dynamic analysis. The former involves examining app code without executing it, and therefore only provides information about *potential* behaviors without evidence that they actually occur in practice. Consequently, static analysis is prone to false positives due to detection of suspicious code branches that may never get executed in practice. Dynamic analysis, on the other hand, involves executing applications in instrumented environments to directly observe their behaviors. Previous approaches to dynamic analysis have involved taint tracking (*i.e.*, modifying data so that it can be observed traversing through a system) or using higher level code instrumentation (*e.g.*, modifying the operating system) [28, 33].

Our approach is based on using instrumented environments [25, 88], because taint analysis is susceptible to control flow attacks and is known to have considerable performance costs and incompatibility issues with a variety of apps, hindering successful execution of real world apps [9, 16, 75].

Figure 2 illustrates our dynamic testing pipeline and its three main components:

1. The testbed retrieves apps to be tested from our corpus of free apps downloaded from Google Play.
2. An app is pushed to a Nexus 5X phone running a bespoke version of Android with our instrumentation. That app is then executed for 10 minutes using simulated user input.

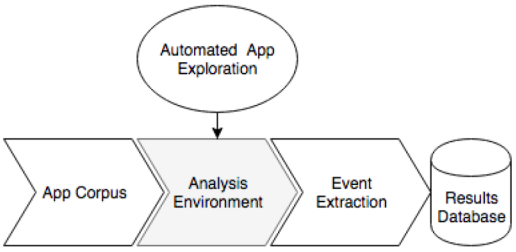

Fig. 2. Dynamic analysis pipeline components.

3. Information about the app's access to sensitive user data and transmissions to third parties are stored in a database for later analysis.

In the remainder of this section, we explain these components in detail, with a particular focus on how we used the system to assess apps for COPPA compliance.

3.1 App Corpus

We scraped the Google Play Store for free apps to test from November 2016 through March 2018. Our scraping strategy focused on the "Top Charts" in each of the Play Store's app categories. Once an app was added to our corpus, we rechecked for new versions every two weeks and queued tests for new releases as they became available. This resulted in a corpus of over 80,000 apps. However, not all of these apps are governed by COPPA. As we explained in the previous section, we focus on the subset of apps that voluntarily chose to appear under Google Play's "Designed for Families" (DFF) program. An app categorized under DFF means that: (i) the developer specifically indicated to Google that the app's intended audience includes, among others, children under 13; (ii) the developer received guidance from Google on COPPA compliance; and (iii) the developer affirmed that the app was in compliance with that guidance. We limit our analysis to the 5,855 tested children's apps that we identified as participating in the DFF program.

Our test corpus of 5,855 apps had a cumulative install count of 4.5B (an average of over 750K installs/app, based on the lower bound of the install range shown in the Play Store). These apps spanned 63 different categories in the Play Store, with the most popular categories being Casual Games, Brain Games, and Educational Games—60% of all the tested apps were from one of these categories. A total of 1,889 unique developers were responsible for all the apps.

3.2 Analysis Environment

Our dynamic analysis focuses on two aspects: how apps access sensitive data and with whom they share it. The former is achieved through a custom version of Android, while the latter is achieved through a custom VPN service, which acts as a localhost man-in-the-middle proxy.

In our custom Android platform (based on v6.0.1), we modified Android's permission system to enable the real-time monitoring of apps' access to protected resources (*e.g.*, location data, address book contacts, *etc.*). We instrumented all the functions in the Android platform that access these sensitive resources (*i.e.*, whenever an app accesses a permission-protected resource, the instrumentation logs the access request). By building this capability into the Android platform, we can observe any Android app without modifying the app itself.

Our framework also includes a modified version of Lumen [67], a network monitoring tool that captures all network traffic generated by the app being tested. Lumen leverages Android's VPN API to redirect all the device's network traffic through a localhost service that inspects all network traffic, regardless of the protocol used, through deep-packet inspection. Lumen installs a root certificate in Android's trusted store so it can also analyze communications protected by TLS (certificate pinning notwithstanding) [65].

While there have been some previous attempts at monitoring resource usage and data sharing in the wild [1, 67, 69, 77, 88, 92], we believe that ours is the first end-to-end analysis platform that can automatically monitor when data is first accessed and where it is ultimately sent.

3.3 Automated App Exploration

Since our analysis framework is based on dynamic analysis, apps must be executed so that our instrumentation can monitor their behaviors. Ideally, our testbed would explore the same code paths that would be triggered when apps are used normally.

We use Android's UI/Application Exerciser Monkey (the "Monkey") [38]—a tool provided by Android's development SDK—to automate and parallelize the execution of apps by simulating user inputs. The Monkey injects a pseudorandom stream of simulated user input events into the app, thereby simulating random UI interactions; it essentially "fuzzes" an app's UI. Because this pseudorandom input is generated from a random seed, it is also reproducible.

Our testing pipeline schedules each app to run for 10 minutes on a Nexus 5X, with the Monkey generating input events during this time period. After each 10 minute execution slot, logs are generated based on the observed behaviors. After each execution, the device goes through a cleaning phase to isolate each test run from one another. In the current setup, we can analyze approximately 1,000 apps/day on 8 phones.

One obvious question regarding the Monkey is how well it is able to uncover the same app functionality that a real user might encounter [2]. Unlike real users, a pseudorandom input generator does not process app visual cues. For example, it does not immediately know that it needs to click a button to dismiss a dialog. This might result in sub-optimal execution path coverage. Therefore, the evaluation presented in this paper is a lower-bound of what an app can do while interacting with a human user: more potential violations are possible due to the execution paths unexplored by the Monkey.

To better understand the effectiveness of the Monkey, we compared its performance to that of a human user. We evaluated it both in terms of the number of Android "Activities" uncovered—unique screens within an app—as well as the number of data flows recorded. We instructed our human tester to explore each app for 10 minutes and to manipulate all interactive elements. Similarly, we configured the Monkey to test each app for 10 minutes, producing a random input every second. We used the Monkey's built-in options to constrain its exploration to the app being tested.

We performed this parallel testing on an initial corpus of 401 apps in December 2016. When comparing the coverage of each method, the human tester missed 9% of the Activities that the Monkey uncovered, whereas the Monkey missed 39% of the Activities that the human uncovered. That is, the Monkey matched or exceeded the human's app screen coverage 61% of the time. In terms of network flows, the human and Monkey testers missed 20% and 34%, respectively. Based on this analysis, we concluded that the Monkey may incur false negatives (*i.e.*, not detecting potential privacy violations), but any potential privacy violations uncovered in our testing environment are observations of *actual* app behaviors, so it does not generate false positives. Therefore, the results produced by our method represent a lower bound of potential COPPA violations.

3.4 Post Processing

The final stage of the pipeline is to analyze the log files generated by our framework. These logs contain both observed permission requests and network flows. We parse these logs by searching them for custom string queries depending on the type of data.

For permission requests, we search the permission log for output from our instrumentation, which records the type and context under which guarded resources are accessed. For network flows, we search for identifiers as string values associated to the particular testing device, such as the phone number, IMEI, MAC address, location, *etc.*, which are listed explicitly alongside each log file at the time we perform our experiment. We only report leaks of these identifiers if we find any of them in an outgoing flow emitted by the tested app.

Because we wrote the instrumentation that monitors guarded resource accesses, detecting these accesses is straightforward. Finding PII inside network transmissions, however, is a greater challenge because different apps and SDKs use different encodings to transmit PII.

Our approach to decoding obfuscated network flows is to find packets that appear to be sending some meaningful data, but for which we were not identifying any PII. We manually inspect and decode this traffic, and then create regular expressions to automatically decode all similarly encoded traffic in the future. Aside from standard encodings, such as base64, URL encoding, and HTML character encoding, we also search for permutations of identifiers: upper and lower cases, strings and binary representations, as well as the values resulting from the MD5, SHA1, and SHA256 hashing algorithms.

We observe that these hash functions are intended to be non-reversible, but the resulting value remains a unique identifier with the same persistent properties as the original, meaning that its suitability for tracking remains the same. Moreover, a brute-force search to reverse a hash value is easily feasible for many identifiers simply because their domain is insufficiently large. Examples of identifiers with a small domain include serial numbers, IMEIs, and phone numbers.

Additionally, for a handful of advertising and analytics SDKs, we also reversed their use of bespoke encodings to transmit PII. For example, ironSource uses AES/CBC with a fixed encryption key (embedded in the SDK) to transmit an initial request for configuration options that includes the advertising ID and sometimes other identifiers, including the e-mail address and IMEI. They use this mechanism despite the use of TLS. StartApp sends location data (when available to the app), as well as MAC addresses and other PII using a Vigenère-style XOR-based encoding using the mask "ENCRYPTIONKEY" and a leetspeak mutation of their company name. Flurry rounds the location coordinates to three decimals and then sends the binary encoding of the resulting floating-point number. ComScore sends the MD5 hash of the hardware serial number by first prefixing it with a per-developer secret ID. Two developers, Disney and Turner, comprise the majority of apps using comScore in our dataset, and so we reversed the secret ID for both developers, compute the resulting hashed serial number, and categorize any transmissions of it as a transmission of the serial number.

PII	Description
AAID	Android Advertising ID
Android ID	Unique ID created at Android setup
GSF ID	Google Services Framework ID
HW ID	Phone hardware ID (serial number)
IMEI	Mobile phone equipment ID
SIM ID	SIM card ID
MAC Address	MAC address of WiFi interface
Email	Email address of phone owner
Phone #	Mobile phone's number
Latitude, Longitude	User location
Router MAC Address	MAC addresses of nearby hotspots
Router SSID	SSIDs of nearby hotspots

Table 1. The types of personal information that are detected in our analysis logs. The top half of the table lists persistent IDs.

Table 1 lists the types of information that we search for in our log files. The top half of the table lists specific identifiers. The Android Advertising ID (AAID) was created by Google as a compromise between allowing user tracking (*e.g.*, for behavioral advertising and profiling) and giving users more control over their privacy: users can reset this identifier or check a box to prevent long-term tracking (the latter option causes a "do not track" flag to be transmitted). Of course, if this identifier is collected alongside other persistent identifiers, this would undermine the intended privacy protections. To prevent this, Google's terms of use indicate that "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" [40]. The other identifiers we examine are either hardware-based and cannot be reset (*e.g.*,, IMEI, MAC address) or cannot be reset *easily*: the Android ID can only be changed via a factory reset, the GSF ID can be reset by creating a new Google account, and the SIM ID can be reset by replacing the SIM card.

As geolocation coordinates are numerical values, we detect the presence of geolocation data in network flows by identifying the latitude and longitude as numbers written out as a string in decimal that matches the integer component and at least the first three decimal values. We also search for the precise latitude and longitude written as a floating-point number and in binary, as well as those values rounded to 3, 4, and 5 decimal places. We require that both the latitude and the longitude appear in the same packet for our instrumentation to consider it a transmission of location. This degree of precision means that our location information was sent with 100 meters of accuracy—well within COPPA's standard of street-level accuracy [22].

4 Analysis

We performed automated analysis on 5,855 Android apps that agree to abide by COPPA as part of their inclusion in the Play Store's Designed for Families (DFF) program. Of these 5,855 apps, 28% accessed sensitive data protected by Android permissions. We also observed that 73% of the tested applications transmitted sensitive data over the Internet.[7] While accessing a sensitive resource or sharing it over the Internet does not necessarily mean that an app is in violation of COPPA, none of these apps attained verifiable parental consent: if the Monkey was able to trigger the functionality, then a child would as well. This suggests that many potential violations are likely occurring, which we discuss in the remainder of this paper: we examine access to personally-identifiable information, sharing of persistent identifiers, the timing of when data is transmitted, and the effectiveness of the Safe Harbor programs.

4.1 Personal Information

In this section, we present our results regarding apps' use of geolocation and contact information. From the 5,855 applications tested, we found: 256 apps (4.4% of 5,855) collecting geolocation data or data sufficient to infer it; 107 sharing the device owner's email address; and 10 sharing the phone number.

7 Some of the COPPA-governed resources are not controlled by Android permissions (*e.g.*, access to many of the persistent identifiers), which is why we observed many more examples of data exfiltration than access to permission-protected resources.

4.1.1 Geolocation via Location APIs

Geolocation data not only reveals where individuals live, but could also enable inferences about their socioeconomic classes, everyday habits, and health conditions, among others [20]. Such inferences could carry life-long implications for children. The 2013 revision to COPPA was in part motivated by the widespread availability of geolocation-enabled mobile apps for children. Unlike other types of identifiers that have exemptions to COPPA's consent requirements for performing activities like "contextual advertising" or "giving notice" [84], *any* access to geolocation information requires *verifiable* parental consent. That the Monkey was able to trigger this functionality with random taps and swipes implies that verifiable parental consent is not being obtained.

Of the 5,855 apps analyzed during the study period, 706 declared either the ACCESS_FINE_LOCATION or ACCESS_COARSE_LOCATION permissions in their manifests, which means that they—and their bundled third-party libraries—could potentially access location data. Our instrumentation observed 235 apps (4.0% of 5,855) actually accessing this data by calling Android location APIs that reveal GPS coordinates. These apps had a cumulative install count of 172M (an average of 734K).

Given the lack of verifiable parental consent, just accessing this data appears to be a potential violation, based on the FTC's guidance [84]. Furthermore, 184 of these apps also transmitted the location data, sharing it with a median of 3 unique domains. A total of 107 unique domains received location data from these apps. The most popular destinations were: *mopub.com* (85 apps), *aerserv.com* (84 apps), *skydeo.com* (80 apps), *youapp.com* (80 apps), and *inner-active.mobi* (76 apps).

One particularly egregious example is app developer TinyLab. We observed that 81 of their 82 apps that we tested shared GPS coordinates with advertisers. Especially popular apps included:

- **Fun Kid Racing** (v3.12, 10-50M installs): GPS data shared with `ads.aerserv.com` (non-TLS), `location-api.skydeo.com`, and `sdk.youappi.com`
- **Fun Kid Racing–Motocross** (v3.12, 10-50M installs): GPS data shared with `ads.aerserv.com` (non-TLS), `location-api.skydeo.com`, `sdk.youappi.com`, and `sdkng.youappi.com`
- **Motocross Kids–Winter Sports** (v3.15, 5-10M installs): GPS data shared with `wv.inner-active.mobi` (non-TLS), `c.adsymptotic.com` (non-TLS), `sdk.youappi.com`, and `location-api.skydeo.com`

Many of the companies receiving location data are advertising firms whose business models rely on user profiling to perform behavioral advertising, which is explicitly prohibited by COPPA. It is particularly important to note that MoPub, the most popular destination for location data among children's apps, clearly states in their terms of service that their service should not be used by any app that collects data from anyone under 13 [61], likely because their privacy policy explicitly states that collected data may be used for behavioral advertising. We discuss the use of prohibited libraries in more detail in Section 4.3.1.

4.1.2 Wi-Fi Router Geolocation

As an alternative to querying a phone's GPS hardware, apps can retrieve the MAC address of the currently-connected Wi-Fi hotspot to infer a user's location with street-level precision. This is because Wi-Fi hotspots tend to have fixed locations and MAC addresses that uniquely identify them. Developers and tracking services can use Wi-Fi-based geocoding services, such as the Google Maps Geolocation API [37] or WiGLE.net [87], to map these MAC addresses to GPS coordinates.

This technique allows app developers to determine the user's location without explicitly asking for the location permission or triggering the location notification icon. The FTC has been pursuing companies engaging in this deceptive practice. For instance, in 2016, they reached a $1M settlement with InMobi over this [23].

To identify children's apps that potentially engage in Wi-Fi geolocation, we searched network flows for MAC addresses and SSIDs of the currently-connected Wi-Fi router. We observed 101 children's apps sharing Wi-Fi router MAC addresses with third parties. The most common recipients were: *greedygame.com* (61 apps), *startappservice.com* (60 apps), *startappexchange.com* (57 apps), *kochava.com* (30 apps), and *appnxt.net* (13 apps). Example apps include:

- **Yousician's "Guitar Tuner Free—GuitarTuna"** (v4.3.6, 10–50M installs): Wi-Fi router MAC transmitted to `control.kochava.com`
- **TabTale's "Pop Girls–High School Band"** (v1.1.9, 1–5M installs): Wi-FI router MAC transmitted to `init.startappservice.com`

Although not as distinct as Wi-Fi router MAC addresses, human-readable network names (SSIDs) can still allow some inferences about users' locations, especially when collected over time and across locations. Retrieving the names of saved networks does not require an app to hold location privileges either. Because string searching for SSID names is prone to false positives, we manually verified that the SSID values we discovered were indeed valid transmissions. We found 148 apps engaging in this behavior, including:

- **Disney's "Where's My Water? Free"** (v1.10.0, 100–500M installs): Wi-FI router name transmitted to `control.kochava.com`
- **Tiny Lab's "Motocross Kids–Winter Sports"** (v2.4.2, 10–50M installs): Wi-FI router name transmitted to `api.greedygame.com`

4.1.3 Contact Information

Android API access to contact information is protected by two different permissions: GET_ACCOUNTS can identify email addresses and other accounts (*e.g.*, Twitter username) accessed by the device, and READ_PHONE_STATE can read the device's phone number. Out of the 5,855 applications we tested, 775 declared the GET_ACCOUNTS permission, which means that 13% *could* access contact information. Subsequently, we observed 254 apps (4.3%) *actually* accessing this information during testing. Similarly, 1,780 applications declared the READ_PHONE_STATE permission, and we observed 634 (10.8%) actually accessing information protected by this permission. The mere collection of this information may not be a violation of the law by itself, since there are several limited exemptions for collecting contact information without first attaining verifiable parental consent [84].

However, the transmission of contact information, in particular to third parties, may be more indicative of a potential COPPA violation. Our testing found 107 children's apps that transmitted the device owner's email address to remote servers. The five most common destinations were: *appspot.com* (79 apps), *tinylabproductions.com* (19 apps), *google.com* (10 apps), *skydeo.com* (5 apps), and *drgames.fr* (3 apps). The transmission of phone numbers was less common: just 10 out of the 5,855 children's apps shared this data with remote services. The following domains received phone numbers: *drgames.fr* (3 apps), *cafe24.com* (2 apps), *oneaudience.com* (2 apps), *gameloft.com* (1 apps), and *mamabearapp.com* (1 app).

4.2 Insecure Transmissions

COPPA requires that children's apps "must establish and maintain reasonable procedures to protect the confidentiality, security, and integrity of personal information collected from children."[8] To examine apps' compliance with this clause, we examined the usage of TLS, when transmitting any type of personal information (including persistent identifiers). Overall, we observed that 2,344 DFF apps do *not* use TLS in at least one transmission containing identifiers or other sensitive information (40.0% of 5,855 apps analyzed). This number also likely represents an upper bound on usage of reasonable procedures to transmit personal information, because we did not examine whether apps that do use TLS are doing so correctly (*e.g.*, proper validation) [65]. Given that TLS is the standard method for securely transmitting information, it could be argued that almost half the apps we examined are not taking "reasonable procedures to protect the confidentiality, security, and integrity of personal information collected from children."

4.3 Persistent Identifiers

Persistent identifiers in Android can be used to identify devices and their users over time and across different services (Table 1). Using persistent identifiers without verifiable parental consent is allowed under COPPA, as long as the identifiers are being used for "internal operations," which the Rule has defined to include "serv[ing] contextual advertisements" [84]. Contextual advertisements refer to ads that are targeted only based on the type of app or service being used, without regard for the user's behaviors or previous interactions: behavioral advertising and user profiling are explicitly prohibited.

Definitively determining whether a persistent identifier is being used to serve contextual versus behavioral ads is likely impossible, as it would require knowing a third party's internal business practices.

However, by examining which identifiers are being shared with which remote services, we can identify *potential* violations. For instance, when identifiers are shared with companies whose primary business models are to perform user profiling or behavioral advertising, and those companies explicitly prohibit the use of their SDKs in children's apps, that would suggest that the identifiers are being used for COPPA-prohibited purposes. Similarly, many third-party SDKs include documentation that explains how app developers can make use of certain configuration options to disable user profiling and behavioral advertising. In this section, we identify apps that use third-party SDKs that are potentially violating COPPA based on their privacy policies and terms of service, we examine whether app developers make proper use of configuration options designed to prevent COPPA violations, and then we examine prohibited uses of the Android Advertising ID (AAID).

Service Name	App Count	Total Installs	Terms of Service
Crashlytics	300	556M	[39]
Supersonic / ironSource	466	481M	[47]
Tapjoy	101	386M	[78]
MoPub	148	296M	[61]
Inneractive	180	239M	[46]
Heyzap	263	150M	[44]
Appnext	46	55M	[6]
Amplitude	17	52M	[4]
Branch	19	42M	[12]
Appboy	1	10K	[5]

Table 2. Most popular third-party service bundled in apps targeting children under 13, whose terms of service prohibit their service from being used in children's apps.

4.3.1 Verboten SDKs

We examined the most popular third parties receiving data from the children's apps that we tested, and identified several that specifically prohibit developers from using these services in children's apps. Excerpts from these agreements are included in Appendix A. Some services' privacy policies explicitly state that collected data may be used for COPPA-prohibited purposes. For instance, MoPub's privacy policy specifically mentions that data may be used "to infer interests and serve ads to users based on their app and website activity and inferred interests" [60].

Table 2 lists the third-party SDKs whose terms of service prohibit their use in children's apps. The table shows the number of apps in our dataset using each SDK, the total number of installs of those apps, and a reference to the SDK's terms of service. Overall, we observed that 1,100 (18.8% of 5,855) apps transmitted persistent identifiers to at least one of these third parties. These apps have collectively seen ∼1.6B installs. Some examples of popular apps doing this include:

8 16 C.F.R. §312.8.

- **Duolingo** (v3.58.0, 100-500M installs): AAID, Android ID, and serial number (HW ID) are sent to `reports.crashlytics.com`
- **Gameloft's Minion Rush** (v2.7.5b, 100-500M installs): Android ID sent to `content.tapjoy.com`, `connect.tapjoy.com`, and `ws.tapjoyads.com`
- **Disney's Where's My Water** (v1.10.0, 100-500M installs): AAID sent to `ads.mopub.com`

While we cannot definitively know whether or not these third parties are using this information for COPPA-prohibited practices, such as behavioral advertising, their terms of service and privacy policies suggest that violations are likely.

4.3.2 Client-Side COPPA Options

Some third-party SDKs offer client-side options—SDK methods and parameters that can be invoked by app developers in their app code—that allow developers to configure the SDKs to be COPPA-compliant. To use these, developers will often pass parameters when initializing the SDK. Later, when host apps communicate with the SDK's servers, they transmit flags alongside the persistent identifiers to indicate that tracking and behavioral advertising should be disabled for that particular user (or ideally, these options may result in apps not sending certain identifiers altogether). Our testing environment is able to detect the presence of these options in outgoing traffic. Of course, to detect a particular SDK's client-side options, we first need to know the format in which these flags appear in traffic to the server (*e.g.*, as keys within JSON objects, HTTP GET variables, *etc.*). This involves reading SDK documentation to understand each SDK's available configuration options, but once known, we can then search our entire results database for the presence of that SDK's flags across any number of apps.

For example, the Facebook API, a service providing social media integration along with targeted advertisements and audience insights, includes a flag in ad requests indicating if the host application is a children's app [29]: The HTTP request to Facebook's servers includes the parameter "`...&COPPA=true&...`"

Unfortunately, few developers appear to use these configuration options. In the case of the Facebook API, this flag is not consistently set to "true" across our corpus of children's apps, even among apps by the same developer: Libii's "Princess Libby & Vampire Princess Bella" (v1.2, 1M installs) sends "`COPPA=true`," while the same developer's "Princess Salon: Frozen Party" (v1.2, 10M installs) sends "`COPPA=false`." We detected the Facebook API being used in 1,280 children's apps we tested (21.9% of 5,855): 342 have the COPPA value set to "`false`," 75 send the value as "`true`," 27 are inconsistent within single tests or across versions, and 836 never send this flag. If we take `COPPA=false` or unset as potential violations, then this suggests that 92% of the apps that use the Facebook API may be using it for COPPA-prohibited activities (whether intentionally or not).

This is not an isolated example, as we have seen this in other apps and SDKs: 57 apps—of which 56 are by Tiny Lab—sent COPPA flags to Mediabrix, with 56 sending "`coppa:true`" and 1 sending "`coppa:false`"; and 76 apps received ad data from Fyber, an ad mediator, containing GET requests to `angsrvr.com`, of which 19 had "`ang_coppa=yes`" and 57 "`ang_coppa=no`."

In our corpus, 12 apps transmitted data to Upsight, who also offers developers configuration options to be COPPA compliant. Of these, only one app transmitted the `opt_out=1` or `opt_out=true` flag [79], while 9 others all set the flag to either "`0`" or "`false`," indicating user tracking would occur (the remaining 2 had inconsistent flags). The 9 apps with "`0`" or "`false`" are reported to be installed on over 19M devices in the U.S.

We also observed 318 children's apps transmitting data to Kochava, which is an "attribution provider." Attribution providers measure the success of ad campaigns that promote other apps by tracking whether users ultimately install an advertised app. It is not clear how this process is compliant with COPPA, given that it involves tracking user behavior (going well beyond the mere serving of contextual ads), and it is unclear how it is *necessary* under the "internal operations" exemptions we listed in Section 2. Nonetheless, Kochava offers an opt-out flag so that app developers can limit the tracking of their app's users (`app_limit_tracking=true/false`). We observed 43 children's apps transmit this flag with a value of "`false`," whereas the remaining 275 did not transmit the flag at all. We did, however, observe two instances of apps limiting user tracking with this flag when looking beyond our corpus of children's apps.

4.3.3 Server-Side COPPA Options

Some SDKs require app developers to use server-side options to indicate that children are among their apps' users: the developer visits a website—such as an online configuration dashboard—and selects an option to indicate that COPPA-prohibited behaviors should be dis-

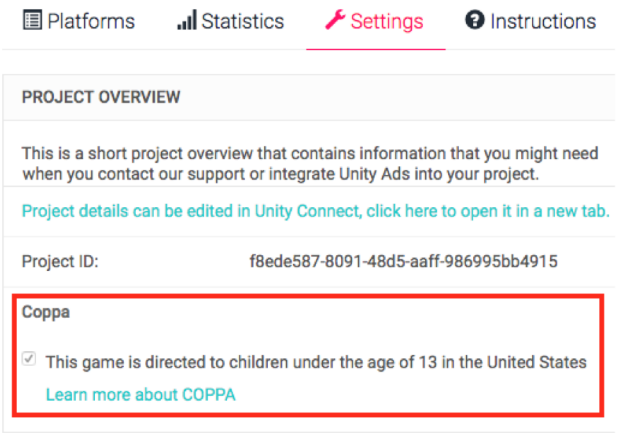

Fig. 3. Unity 3D server-side configuration options to opt-out of COPPA-prohibited tracking/advertising behaviors.

abled. In most cases, the SDK still transmits the same data that it normally would for non-child-directed apps, but the server is now set to disregard identifiers or otherwise not use the collected data for prohibited purposes. In these cases, outgoing network traffic cannot be used to detect COPPA-compliant settings.

We observed, however, that some services will actually transmit COPPA configuration flags *back* to the app to tell it that the bundled SDK should switch to a COPPA-compliant mode. We used our network monitoring tools to determine which SDKs sent these options back to apps, which allowed us to detect which children's apps did or did not receive these incoming flags—suggesting those apps' developers may have failed to select COPPA-compliant options.

We detected this behavior in Unity 3D, by far the most popular development platform (which offers revenue and analytics modules) among the apps in our corpus: we identified it in 2,909 children's apps (49.7% of 5,855). Of these, we observed 1,068 receive a flag, "`coppaCompliant`," which was included as part of an inbound JSON configuration object from Unity's servers. However, only 479 (out of 1,068) have this flag set to "`true`" (COPPA compliant), whereas 589 have it set to `false`. This suggests that potentially the majority of children's apps with the Unity 3D SDK may not be using available COPPA options properly or at all (83.5% of 2,909 do not receive a "`coppaCompliant=true`" flag).

4.3.4 Android Advertising ID

Google introduced the user-resettable Android Advertising ID (AAID) in 2014 in order to give end-users better control over how online services track them over time and across devices. Unlike the other persistent identifiers available on Android, users can generate new AAIDs without resorting to drastic measures, like performing a factory reset or replacing the device. In their terms of service, Google requires developers and SDKs to use the AAID as the only persistent identifier for advertising purposes, in lieu of all other identifiers provided by the device's hardware or software [40]. As we explained earlier, transmitting the AAID alongside non-resettable persistent identifiers would completely negate the privacy-preserving properties of the AAID.

Domain	Apps that send IDs	Apps sending non-AAID IDs	Compliant w/ Google policy
doubleclick.net	168	1	99%
lkqd.net	65	1	98%
mopub.com	148	3	98%
liftoff.io	178	5	97%
applifier.com	684	27	96%
vungle.com	975	115	88%
kochava.com	161	30	81%
inmobi.com	153	29	81%
adcolony.com	557	108	80%
supersonicads.com	465	144	69%
tapjoy.com	98	96	2%
tapjoyads.com	95	94	1%
chartboost.com	859	858	$< 1\%$
greedygame.com	59	59	$< 1\%$

Table 3. Ad domains collecting any persistent identifier from apps vs. any non-Advertising-ID identifier.

In order to measure how consistently advertisers follow this requirement, we compared the total number of apps that send any kind of persistent identifier to a given domain against the number of apps that send any *non-AAID* identifier to the same domain; this reveals which services are using *only* the AAID (Table 3).

Out of all the third-party domains we observed, ad domains were identified by visiting the websites and finding keywords such as "monetization," "marketing," "mobile advertisement," "promotion," and "ad platform." For ad domains contacted by least 50 apps, we found a sharp divide between ad domains that receive only the AAID from a majority of apps that contact them (*i.e.*, at least 69% compliance with policy) vs. domains that appear to not use the AAID at all, favoring non-resettable identifiers instead (*i.e.*, at most 2% compliance). Even though the AAID has been available since 2014, and even though Designed for Families apps are supposed to be held to a higher standard when it comes to advertising practices, there are still advertising networks used in children's apps that continue to collect immutable identifiers that could enable behavioral ad-

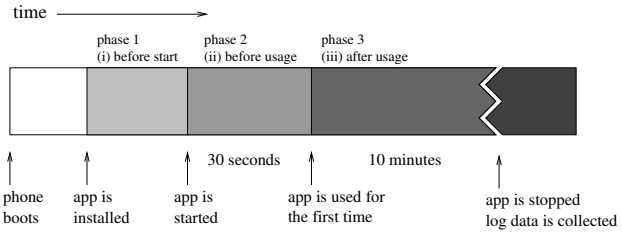

Fig. 4. Timeline for Monkey runs. (i) The time when the app is installed but before it is started. (ii) The time after the app is started, but before the Monkey starts interacting with the app. (iii) The time when the Monkey is interacting with the app.

vertising and long-term tracking. All in all, we observed 2,281 children's apps transmit the AAID alongside another more persistent identifier to the same destination. This represents approximately 39% of our corpus that does not appear to follow Google's terms of service.

4.4 Timing of Data Exfiltration

During initial testing, we observed that many apps transmitted PII immediately upon launch—before any user interaction whatsoever. Because this happens immediately, there is no opportunity for verifiable parental consent (or rather, any type of consent). Transmitting PII without interaction also eases concerns that the automated Monkey testing is missing important code paths that lead to transmissions.

To quantify the degree to which this is happening, we measured the prevalance of apps connecting to destinations prior to any user input. Figure 4 illustrates how we divide outgoing transmissions into one of three non-overlapping phases: phase 1 begins after installing the app, phase 2 begins after starting the app, and phase 3 begins after interacting with the app. Our system intentionally provides a 30-second delay before providing input events, and so we use the first 25 seconds as a safe bound to categorize transmissions in phase 2. Phase 3 begins afterwards and lasts about ten minutes.

As expected, we found no PII transmissions occuring in phase 1. We therefore categorize the remaining transmissions as either *never occuring* in phase 2 or *occuring* in phase 2. That is, an app may send the location repeatedly while running, but we consider this transmission to occur in phase 2 provided we observe at least one transmission prior to any interaction and therefore without any opportunity for parental consent. Out of the apps that send PII, only 13% can be said to *never* send PII in phase 2, the rest had some version that sent PII before any consent was possible.

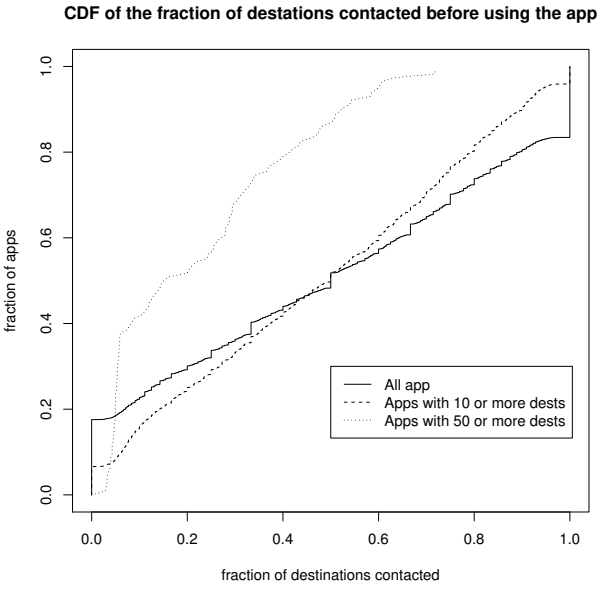

Fig. 5. Distribution of apps as measured by the fraction of domains that they first contact during the time after the app is started, but before any user interaction has occurred. We characterize all apps, as well as the subset of apps that contact at least 10 destinations and 50 destinations.

Additionally, we examined how children's apps connected to remote services during phase 2. Figure 5 illustrates the distribution of apps measured by the fraction of remote domains they contact just by starting the app (out of all the domains they contact). The Y-axis represents the space of all apps, and the X-axis represents the space of domains. As many of our corpus apps have multiple versions, we use results from the most-recent version for each app.

We found that 5,296 children's apps (*i.e.*, over 90% of our corpus) contact remote destinations. Of those, 2,858 contact at least 10 different destinations, and 114 that contact at least 50. Figure 5 tells us that nearly half of the apps contact half their destinations in phase 2. That is, these apps do so *on their own* prior to providing notice or obtaining consent.

4.5 Safe Harbor

The COPPA rules specify a "Safe Harbor" program that allows companies to submit the children's apps and websites they develop for review by a designated organization. This organization then evaluates the app or website for COPPA compliance. Under the program, any company that receives approval under the Safe Harbor program is less directly subject to FTC enforcement ac-

tions. The goal of the program is to allow companies to implement self-regulatory guidelines that provide "the same or greater protections for children," as compared to what is required under COPPA [82]. Operators under a Safe Harbor program are subjected to more lenient enforcement procedures, prior to any formal enforcement from the FTC. As of March 2018, the FTC has designated seven companies as Safe Harbor providers [81]:

- Integrity (Aristotle International, Inc.)
- Children's Advertising Review Unit (CARU)
- Entertainment Software Rating Board (ESRB)
- iKeepSafe
- kidSAFE
- Privacy Vaults Online, Inc. (d/b/a PRIVO)
- TRUSTe

We examined whether apps certified by these seven organizations exhibit better behavior with regards to the collection and sharing of personal information than our full set of children's apps.

Identifying certified apps is not straightforward: with the exception of kidSAFE [74] and CARU [18], none of these organizations list the apps or developers that they have certified (some organizations certify a developer's practices, and therefore all apps by that developer are implicitly certified, whereas others certify apps individually). As a result, we performed several web searches to find companies that had either listed the certifying organizations in their privacy policies, displayed their seals online, or via searches for the verification URLs. For instance, developers certified by ESRB post a seal on their websites that leads to a verification URL of the form, `http://www.esrb.org/confirm/*.aspx`. By searching Google and Archive.org for URLs of this form (*e.g.*, on Google, `site:http://www.esrb.org/confirm/`), we were able to identify 43 apps certified by ESRB. Using these techniques, we ultimately found 237 free children's apps for Android across the 7 Safe Harbor organizations.

Our sample size of 237 may seem small, especially when considering the number certified by each organization individually. However, all available evidence suggests that few apps are certified by these services. For instance, TRUSTe's CEO disclosed in April 2017 that they have only certified 20 companies [8]. The 24 apps that we were able to associate with TRUSTe certification represent 7 of those companies. This is likely representative, as this corresponds to 35% of all the companies that TRUSTe has claimed to have ever certified for COPPA compliance. It is also likely that the remaining 13 companies only developed child-directed websites or iOS apps, and not apps for the Android platform; several other companies' privacy policies stated that their TRUSTe seals only apply to their websites, whereas we also found several TRUSTe-certified companies that developed iOS games with no Android counterparts.

Name (n)	*Loc./Contact*		*Identifiers*		*non-TLS*	
Integrity (1)	-		-		-	
CARU (66)	1	(1.5%)	38	(57.6%)	25	(37.9%)
ESRB (43)	1	(2.3%)	27	(62.8%)	12	(27.9%)
iKeepSafe (4)	1	(25.0%)	2	(50.0%)	-	
kidSAFE (42)	4	(9.5%)	29	(69.0%)	6	(14.3%)
PRIVO (57)	7	(12.3%)	44	(77.2%)	23	(40.4%)
TRUSTe (24)	10	(41.7%)	16	(66.7%)	11	(45.8%)

Table 4. List of COPPA Safe Harbor organizations and the number of certified apps from each that we were able to analyze. The other columns enumerate the number of apps transmitting location or contact information (phone number or email address), persistent identifiers, and not using TLS.

Table 4 lists the number of certified apps that we found, along with their respective certifying organizations. Overall, given that the Safe Harbor program aims to create privacy practices that go above and beyond COPPA's minimum requirements, we were surprised that the privacy practices of these apps—in aggregate—were quite similar to the practices of the other apps in our DFF dataset, which we presented earlier. For instance, 156 apps (65.8% of 237) transmitted persistent identifiers, including 151 (63.7% of 237) that transmitted non-AAID identifiers that may be in violation of Google's policies. Of these, 77 (49.4% of 150) did so without using TLS (*i.e.*, unencrypted). That is to suggest, these apps are arguably not taking "reasonable" precautions to secure personal information [22], which itself may be a potential COPPA violation.

The Safe Harbor apps also used several forbidden third-party SDKs—SDKs whose terms of service prohibit their inclusion in children's apps—that we described in Section 4.3.1. Notably, 78 apps (32.9% of 237) transmitted identifiers to at least one of the forbidden advertising and analytics providers:

- MoPub (31 apps)
- Crashlytics (29 apps)
- Tapjoy (26 apps)
- ironSource (19 apps)
- Branch (9 apps)
- Inneractive (6 apps)
- Amplitude (2 apps)
- Heyzap (2 apps)
- Appboy (1 app)

In terms of transmitting personal information without consent, there is little (or no) difference between the certified apps and the DFF corpus. For instance, eight apps (3.4% of 237 vs. 3.1% of 5,855 in the full corpus) transmit GPS coordinates, 4 transmit email addresses (1.7% of 237 vs. 1.8% of 5,855 in the full corpus), however none transmitted phone numbers. We also observed 15 apps (6.3% of 237 vs. 3.0% of 5,855) gathering the MAC address or SSID of the Wi-Fi hotspot, which could be used to surreptitiously track location.

Overall, these observations corresponded to 24 unique apps (10.1% of 237; double the rate of the DFF corpus) transmitting PII (*i.e.*, location data or contact information) without consent. While some (or all) of these might be exempted under COPPA for various reasons that are not apparent to us, we believe it is important to note that the privacy behaviors of the certified apps are not appreciably better than those of children's apps that have not been certified under Safe Harbor programs (and may be worse). We have listed the details of some of these potential violations in Appendix B.

5 Related Work

In this section, we summarize relevant prior work on performing privacy analyses on mobile apps and on identifying potential COPPA violations.

5.1 Privacy Analysis of Mobile Apps

Prior work has studied how mobile apps access personal data using one of two techniques: static and dynamic analysis. Static analysis evaluates software without actually executing it, instead inspecting the app's binary or source code. Call graph analysis is the most common technique used in static analysis for analyzing the use of sensitive resources [3, 7, 49, 93]. Static analysis techniques, however, do not produce observations of privacy violations. Instead, they only suggest that a violation is possible provided that the code actually executes.

Dynamic analysis is performed *at runtime*, leveraging instrumented code to track sensitive data in memory or to intercept system calls that expose sensitive information. Taint analysis is a popular dynamic analysis technique [28, 33]. However, it can be inefficient and prone to control flow attacks [9, 75]. Higher-level code instrumentation is a better alternative due to readily available system info [25, 88] and is transparent to apps.

Network monitoring is another dynamic analysis technique, which measures how mobile apps interact with remote services. This is valuable, as 99% of PII leaks occur via Internet traffic [52]. Previous work has already identified the main stakeholders in the collection of PII [66] and has characterized prevalent patterns and concerns [68, 69, 77, 86].

While dynamic analysis provides empirical observations of PII leaks occurring under actual use, it requires that the app is actually executed. To help automate this process, researchers have developed tools to simulate user actions [15, 38, 42, 50, 55, 68]. The Android Monkey that we use in this work is one such system.

The research presented in this paper builds on our prior privacy-monitoring tools [67, 70, 89], providing us with an automated, scalable, and holistic view of how apps access and share sensitive data.

5.2 Children's Applications

Previous efforts have studied COPPA and children's apps from various perspectives. Several researchers have analyzed a range of threats to minors in online social media platforms [56, 85], websites [10, 14], and smart toys [58, 90, 91], as well as the appropriateness of in-app ads targeting children [17]. Previous work also examined risks posed by third-party components bundled in children's apps, with a focus on targeted advertisements [11, 53]. Other research has focused on methods aiding developers to make their apps more child-friendly in terms of content and privacy [45, 51].

COPPA requires developers of children's apps to offer privacy policies that clearly explain their data usage and sharing practices. Parsing and properly understanding privacy policies, however, is widely considered a hard problem due to policies' complex legal language [43, 57, 63, 64, 71]. Recent work applied machine learning and static analysis to show that only 36% of apps tested meet COPPA's privacy policy requirements [92]. We compliment this work by focusing on actual app behaviors that are likely to violate COPPA, rather than on the apps' stated privacy policies.

6 Discussion

We present the first large-scale dynamic analysis of children's apps. Our main objective was to deploy an automated system to analyze—at scale—how these apps are

complying with COPPA. We identified several concerning violations and trends: clear violations when apps share location or contact information without consent (4.8%), sharing of personal information without applying reasonable security measures (40.0%), potential noncompliance by sharing persistent identifiers with third parties for prohibited purposes (18.8%), and ignorance or disregard for contractual obligations aimed at protecting children's privacy (39.0%). Overall, roughly 57% of the 5,855 child-directed apps that we analyzed are potentially violating COPPA.

6.1 Preventing Potential Violations

To help parents understand the privacy implications of the apps used by their children, we have made our results available online (https://www.appcensus.mobi/). Beyond parents, however, each stakeholder (*i.e.*, app developers, distribution channels, third-party libraries, and regulators) in this ecosystem could take actions to prevent the potential violations that we identified.

For its part, Google already has taken steps to enforce COPPA compliance: the Designed for Families program presents developers of children's apps with information on COPPA and requires that they certify that they are in compliance. However, as our results show, there appears to not be any (or only limited) enforcement. Google already performs static and dynamic analysis on apps submitted to the Play Store [54], so it should not be hard for them to augment this analysis to detect non-compliant entities.

For instance, despite InMobi being sanctioned over its deceptive collection of location data [23], we observed two children's apps still using a version of the InMobi SDK that continued this practice ("Angry Bunny Race: Jungle Road" by Tiny Lab Productions and "Candy's Airport" by Libii). Similarly, several third-party SDKs prohibit being used in children's apps. However, use of these libraries is rampant (18.8% of 5,855). One of these SDKs, Crashlytics, is even owned by Google. Given Google's infrastructure and internal data on DFF participation, it would likely be trivial for them to detect the use of known non-compliant or prohibited libraries and notify the app developers or take actions.

Similar analysis could be used to enforce Google's own policies. Google prohibits apps from transmitting the privacy-preserving Android Advertising ID (AAID) alongside other persistent identifiers [40]. Yet, as we showed, sending non-AAID identifiers is rampant among Android apps: roughly 39% of the children's apps we tested engage in this practice. Google could do a more active vetting process by using its existing app auditing tools to detect these types of policy violations and to take proper actions—whether internally or reporting them to relevant authorities.

Third-party services could also take measures to prevent COPPA violations. SDKs that prohibit inclusion in child-directed apps receive app names amongst the data flowing to their servers. This implies that these SDK providers have actual data that can reasonably suggest whether their SDKs are being used in child-directed apps. In fact, many of these companies are in the business of collecting app intelligence and therefore have access to data such as app categories and keywords, which can be used to infer whether a given app is indeed designed for children. Using the data they already have, these SDKs could simply deny service to known children's apps or even automatically adjust their data collection to be in compliance with COPPA.

App developers and SDKs currently have a financial incentive to ignore potential violations: limiting data collection or the uses for collected data results in decreased revenues (*i.e.*, behavioral advertising yields higher returns than contextual advertising). Despite these financial incentives, we suspect that many privacy violations are unintentional and caused by misunderstandings of third-party SDKs. Due to technologies like ad mediation, many of the third parties receiving user data are selected dynamically at runtime. For this reason, we suspect that most app developers cannot identify all the third parties who may receive user data from their apps, and are even less likely to understand each SDK's possible COPPA configuration options. Nonetheless, app developers are still liable for the behaviors of SDKs they embed in their apps. Thus, app developers could benefit from our tools by allowing them to test the privacy behaviors of their apps prior to release.

Similarly, our tools could benefit regulators in investigating the market for noncompliance, by making it easier for them to detect violations and bring enforcement actions. If these enforcement actions are brought publicly, it may motivate other app developers to pay more attention to the privacy behaviors of their apps.

While the FTC's Safe Harbor program was created to help developers get their apps certified for compliance, few apps are actually certified. Moreover, we showed that potential violations are prevalent among certified apps. Based on our data, it is not clear that industry self-regulation has resulted in higher privacy standards; some of our data suggest the opposite. Thus, industry self-regulation appears to be ineffective.

6.2 Limitations and Future Work

In this paper, we do not mean to show definitive legal liability, nor do we offer legal advice: we draw attention to *potential* COPPA rule violations, as detected by our methods, designed from our understanding of the law and constrained by technical limitations. COPPA includes language that outlines exemptions to some of its requirements, and so our system is unable to account for the full range of possibilities that those exemptions may cover. Establishing legal liability requires nuanced case-by-case analysis of individual products and practices, which is beyond the scope of this work.

Regarding the ethics of this research, corporations are not people, and so IRB approval was not sought. The Electronic Communications Privacy Act does not apply,[9] because our tools do not analyze or observe human communications. As to the Computer Fraud and Abuse Act (CFAA),[10] our tools merely execute apps as a user would and make no attempt to access restricted content. As to terms of service issues, randomizing program inputs (*i.e.*, the "Monkey") for research is common practice [50], and courts have routinely held that breaches of terms of service are not CFAA violations [26, 27, 80]. Finally, we believe this research is exempt from the Digital Millennium Copyright Act,[11] because it is legitimate security research in the public interest, devices and software were lawfully acquired, and no laws were violated.

We note that the Monkey does not exhaustively execute all code paths in apps. While it does find a number of potential privacy violations, many more may exist. Moreover, some apps have complex UI elements that require precise sequences of inputs to activate. These include sliders, timed events, parental gates, and login screens. Given the Monkey's lack of awareness of what is on the screen, it is unlikely to progress past such elements. Additionally, our current method of inspecting TLS-encrypted traffic cannot decrypt traffic from apps that implement certificate pinning; however, this has only a minor impact on our results, as only 9 services (*e.g.*, Yandex, AppoDeal, and TabTale) support some form of pinning that could evade our traffic analysis.

As such, our results should be taken as a *lower bound* of privacy-relevant events. One opportunity for future work is using static analysis to refine the Monkey, better directing it to the areas of the screen likely to be responsive or trigger the use of a sensitive resource. Another possibility is to augment the Monkey with crowdsourced input from remote human users.

Similarly, while we contend that our approach is much more accurate in detecting violations than static analysis approaches, we believe that these approaches are complementary. In fact, our discovery that StartApp was using a Vigenère-style encryption of location and router MAC address data was due to manual static analysis, which helped to uncover hidden sensitive flows from our stored execution logs.

Our analysis is based on COPPA's requirements, but there are other relevant privacy regulations governing mobile apps that our system could support, such as California's Online Privacy Protection Act (CalOPPA) and the European Union's General Data Protection Regulation (GDPR) and ePrivacy directives. Our framework can be easily extended to other regulations.

Finally, our layered architecture is easily extensible to analyze a much larger quantity of apps. We are in the process of migrating the system to a parallelized virtual setup, enabling us to run hundreds of simultaneous Android instances. This will also enable us to perform longitudinal analyses to examine mobile privacy trends.

6.3 Conclusion

Our work stands apart from prior research due to our system's ability to uncover potential privacy issues prevalent in large numbers of children's apps. Given the number of children's apps and a complex third-party ecosystem, analysis at scale is important to properly understand the privacy landscape. Although we cannot know the true number of children's apps in the Play Store, we believe that our results are representative, given that the apps that we examined represent the most popular free ones.

We believe that this work illustrates the utility of our automated app analysis testbed which, with further development, will have impact on multiple stakeholders. End-users can examine our results to understand the privacy behaviors of the apps they use (or plan to use). Developers can use our testing infrastructure to assess how well their apps comply with their privacy policies and regulatory requirements, prior to releasing those apps to the public. Finally, regulators can use it to detect deceptive and suspicious activities in the marketplace as part of investigations.

9 18 U.S.C. §2510 *et seq.*
10 18 U.S.C. §1030.
11 17 U.S.C. §1201.

Acknowledgments

This research was supported by NSF grants CNS-1318680 and CNS-1564329, the Data Transparency Lab, DHS contract FA8750-16-C-0140, and the Center for Long-Term Cybersecurity (CLTC) at U.C. Berkeley. We would like to thank Jianan Amber Lu and Ehimare Okoyomon for their assistance, Joe Calandrino for feedback, as well as Refjohürs Lykkewe.

References

[1] H. Almuhimedi, F. Schaub, N. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, L. Cranor, and Y. Agarwal. Your Location has been Shared 5,398 Times! A Field Study on Mobile App Privacy Nudging. Technical Report CMU-ISR-14-116, Carnegie Mellon University, 2014.

[2] D. Amalfitano, A. R. Fasolino, and P. Tramontana. A GUI Crawling-Based Technique for Android Mobile Application Testing. In *Proc. of IEEE ICSTW*, 2011.

[3] D. Amalfitano, A. R. Fasolino, P. Tramontana, B. D. Ta, and A. M. Memon. MobiGUITAR: Automated Model-Based Testing of Mobile Apps. *IEEE Software*, 2015.

[4] Amplitude, Inc. Privacy policy. https://amplitude.com/privacy, February 12 2017. Accessed: September 29, 2017.

[5] Appboy, Inc. Terms of Service. https://www.appboy.com/legal/, September 1 2017. Accessed: September 29, 2017.

[6] Appnext Ltd. Terms & conditions – publishers. https://www.appnext.com/terms-conditions/, October 1 2017. Accessed: September 29, 2017.

[7] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie. PScout: Analyzing Android Permission Specification. In *Proc. of ACM CCS*, 2012.

[8] C. Babel. Protecting kids' privacy – an ever-evolving effort. http://www.trustarc.com/blog/2017/04/06/protecting-kids-privacy-ever-evolving-effort/, April 6 2017. Accessed: September 29, 2017.

[9] G. S. Babil, O. Mehani, R. Boreli, and M. A. Kaafar. On the Effectiveness of Dynamic Taint Analysis for Protecting Against Private Information leaks on Android-based Devices. In *Proc. of SECRYPT*, 2013.

[10] F. Bélanger, R. E. Crossler, J. S. Hiller, J. Park, and M. S. Hsiao. Pocket: A tool for protecting children's privacy online. *Decision Support Systems*, 2013.

[11] R. Bhoraskar, S. Han, J. Jeon, T. Azim, S. Chen, J. Jung, S. Nath, R. Wang, and D. Wetherall. Brahmastra: Driving Apps to Test the Security of Third-Party Components. In *USENIX Security Symposium*, 2014.

[12] Branch Metrics, Inc. Terms & policies. https://branch.io/policies/, May 16 2017. Accessed: September 29, 2017.

[13] Buongiorno UK Limited. Privacy. http://www.kidzinmind.com/uk/privacy. Accessed: September 29, 2017.

[14] X. Cai and X. Zhao. Online Advertising on Popular Children's Websites: Structural Features and Privacy Issues. *Computers in Human Behavior*, 2013.

[15] P. Carter, C. Mulliner, M. Lindorfer, W. Robertson, and E. Kirda. CuriousDroid: Automated User Interface Interaction for Android Application Analysis Sandboxes. In *Proc. of FC*, 2016.

[16] L. Cavallaro, P. Saxena, and R. Sekar. On the Limits of Information Flow Techniques for Malware Analysis and Containment. In *Proc. of DIMVA*, pages 143–163. Springer-Verlag, 2008.

[17] Y. Chen, S. Zhu, H. Xu, and Y. Zhou. Children's Exposure to Mobile In-App Advertising: An Analysis of Content Appropriateness. In *Proc. IEEE SocialCom*, 2013.

[18] Children's Advertising Review Unit. Supporters. http://www.caru.org/support/supporters.aspx. Accessed: September 29, 2017.

[19] Class Twist, Inc. Privacy policy. https://www.classdojo.com/privacy/, September 14 2017. Accessed: September 29, 2017.

[20] U.S. Federal Trade Commission. FTC Testifies on Geolocation Privacy. https://www.ftc.gov/news-events/press-releases/2014/06/ftc-testifies-geolocation-privacy. Accessed: September 29, 2017.

[21] U.S. Federal Trade Commission. FTC Warns Children's App Maker BabyBus About Potential COPPA Violations, 2014.

[22] U.S. Federal Trade Commission. Complying with COPPA: Frequently Asked Questionss, 2015.

[23] U.S. Federal Trade Commission. Mobile Advertising Network InMobi Settles FTC Charges It Tracked Hundreds of Millions of Consumers' Locations Without Permission, 2016.

[24] U.S. Federal Trade Commission. Two App Developers Settle FTC Charges They Violated Children's Online Privacy Protection Act. https://www.ftc.gov/news-events/press-releases/2015/12/two-app-developers-settle-ftc-charges-they-violated-childrens, 2016. Accessed: September 26, 2017.

[25] M. Conti, B. Crispo, E. Fernandes, and Y. Zhauniarovich. Crêpe: A system for Enforcing Fine-grained Context-related Policies on Android. *IEEE Transactions on Information Forensics and Security*, 2012.

[26] Electronic Frontier Foundation. United States v. David Nosal. https://www.eff.org/cases/u-s-v-nosal, 2015.

[27] Electronic Privacy Information Center (EPIC). hiQ Labs, Inc. v. LinkedIn Corp. https://epic.org/amicus/cfaa/linkedin/, 2017.

[28] W. Enck, P. Gilbert, B. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *Proc. of USENIX OSDI*, 2010.

[29] Facebook. Coppa an - facebook audience net. https://developers.facebook.com/docs/audience-network/coppa. Accessed: November 30, 2017.

[30] FamilyTime. App privacy policy. https://familytime.io/legal/app-privacy-policy.html, March 28 2015. Accessed: September 29, 2017.

[31] Finny Inc. Privacy policy. https://www.myfinny.com/privacypolicy, March 7 2016. Accessed: September 29, 2017.

[32] Fuel Powered, Inc. Terms of service. https://www.fuelpowered.com/tos, March 23 2017. Accessed: September 29, 2017.

[33] C. Gibler, J. Crussell, J. Erickson, and H. Chen. AndroidLeaks: Automatically Detecting Potential Privacy Leaks in

Android Applications on a Large Scale. In *Proc. of TRUST*. Springer-Verlag, 2012.

[34] Google, Inc. Coppa compliance and child-directed apps | families and coppa - developer policy center. https://play.google.com/about/families/coppa-compliance/. Accessed: November 26, 2017.

[35] Google, Inc. Distribution of android versions. http://developer.android.com/about/dashboards/index.html. Accessed: March 21, 2018.

[36] Google, Inc. Program requirements | families and coppa - developer policy center. https://play.google.com/about/families/designed-for-families/program-requirements/. Accessed: September 26, 2017.

[37] Google, Inc. The Google Maps Geolocation API. https://developers.google.com/maps/documentation/geolocation/intro. Accessed: September 29, 2017.

[38] Google, Inc. UI/Application Exerciser Monkey. https://developer.android.com/tools/help/monkey.html.

[39] Google, Inc. Crashlytics agreement. https://try.crashlytics.com/terms/terms-of-service.pdf, January 27 2017. Accessed: September 29, 2017.

[40] Google, Inc. Usage of android advertising id. https://play.google.com/about/monetization-ads/ads/ad-id/, 2017. Accessed: November 30, 2017.

[41] M. I. Gordon, D. Kim, J. Perkins, Gilhamy, N. Nguyenz, and M. Rinard. Information-Flow Analysis of Android Applications in DroidSafe. In *Proc. of NDSS Symposium*, 2015.

[42] S. Hao, B. Liu, S. Nath, W. G.J. Halfond, and R. Govindan. PUMA: Programmable UI-automation for Large-scale Dynamic Analysis of Mobile Apps. In *Proc. of ACM MobiSys*, 2014.

[43] H. Harkous, K. Fawaz, K. G Shin, and K. Aberer. PriBots: Conversational Privacy with Chatbots. In *Proc. of USENIX SOUPS*, 2016.

[44] Heyzap, Inc. Heyzap sdk. https://www.heyzap.com/legal/heyzap_sdk, April 24 2014. Accessed: September 29, 2017.

[45] B. Hu, B. Liu, N. Z. Gong, D. Kong, and H. Jin. Protecting your Children from Inappropriate Content in Mobile Apps: An Automatic Maturity Rating Framework. In *Proc. of ACM CIKM*, 2015.

[46] Inneractive Ltd. Inneractive general terms. http://inneractive.com/terms-of-use/, September 24 2017. Accessed: September 29, 2017.

[47] ironSource Ltd. Privacy policy. https://www.supersonic.com/privacy-policy/, July 14 2016. Accessed: September 29, 2017.

[48] J. Kim, Y. Yoon, K. Yi, and J. Shin. ScanDal: Static Analyzer for Detecting Privacy Leaks in Android Applications. *IEEE MoST*, 2012.

[49] I. Leontiadis, C. Efstratiou, M. Picone, and C. Mascolo. Don't kill my ads! Balancing Privacy in an Ad-Supported Mobile Application Market. In *Proc. of ACM HotMobile*, 2012.

[50] C. M. Liang, N. D. Lane, N. Brouwers, L. Zhang, B. F. Karlsson, H. Liu, Y. Liu, J. Tang, X. Shan, R. Chandra, and F. Zhao. Caiipa: Automated Large-scale Mobile App Testing Through Contextual Fuzzing. In *Proc. of ACM MobiCom*, New York, NY, USA, 2014.

[51] I. Liccardi, M. Bulger, H. Abelson, D. J. Weitzner, and W. Mackay. Can Apps Play by the COPPA Rules? In *Proc. of IEEE PST*, 2014.

[52] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. van der Veen, and C. Platzer. Andrubis - 1,000,000 Apps Later: A View on Current Android Malware Behaviors. In *Proc. of IEEE BADGERS Workshop*, 2014.

[53] M. Liu, H. Wang, Y. Guo, and J. Hong. Identifying and Analyzing the Privacy of Apps for Kids. In *Proc. of ACM HotMobile*, 2016.

[54] H. Lockheimer. Android and security. http://googlemobile.blogspot.com/2012/02/android-and-security.html, February 2 2012.

[55] A. Machiry, R. Tahiliani, and M. Naik. Dynodroid: An Input Generation System for Android Apps. In *Proc. of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, 2013.

[56] M. Madden, A. Lenhart, S. Cortesi, U. Gasser, M. Duggan, A. Smith, and M. Beaton. Teens, Social Media, and Privacy. *Pew Research Center*, 21:2–86, 2013.

[57] A.K. Massey, J. Eisenstein, A.I. Antón, and P.P. Swire. Automated Text Mining for Requirements Analysis of Policy Documents. In *Proc. of IEEE Requirements Engineering Conference (RE)*, 2013.

[58] E. McReynolds, S. Hubbard, T. Lau, A. Saraf, M. Cakmak, and F. Roesner. Toys That Listen: A Study of Parents, Children, and Internet-Connected Toys. In *Proc. of ACM CHI*, 2017.

[59] Miniclip SA. Miniclip privacy policy. https://www.miniclip.com/games/page/en/privacy-policy/, October 29 2014. Accessed: September 29, 2017.

[60] MoPub Inc. Mopub privacy policy. https://www.mopub.com/legal/privacy/, July 19 2017. Accessed: November 30, 2017.

[61] MoPub Inc. Mopub terms of service. https://www.mopub.com/legal/tos/, August 22 2017. Accessed: September 29, 2017.

[62] NFL Enterprises LLC. Nfl.com privacy policy. http://www.nfl.com/help/privacy, September 15 2017. Accessed: September 29, 2017.

[63] A. Oltramari, D. Piraviperumal, F. Schaub, S. Wilson, S. Cherivirala, T.B. Norton, N.C. Russell, P. Story, J. Reidenberg, and N. Sadeh. PrivOnto: A Semantic Framework for the Analysis of Privacy Policies. *Semantic Web*, (Preprint), 2016.

[64] I. Pollach. What's wrong with online privacy policies? *Commun. ACM*, 50(9):103–108, September 2007.

[65] A. Razaghpanah, A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill. Studying TLS Usage in Android Apps. In *Proc. of ACM CoNEXT*, 2017.

[66] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill. Apps, Trackers, Privacy, and Regulators: A Global Study of the Mobile Tracking Ecosystem. In *Proc. of NDSS Symposium*, 2018.

[67] A. Razaghpanah, N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, P. Gill, M. Allman, and V. Paxson. Haystack: In Situ Mobile Traffic Analysis in User Space. *arXiv preprint arXiv:1510.01419*, 2015.

[68] J. Ren, M. Lindorfer, D. J. Dubois, A. Rao, D. Choffnes, and N. Vallina-Rodriguez. Bug Fixes, Improvements,... and Privacy Leaks. In *In. Proc. of NDSS Symposium*, 2018.

[69] J. Ren, A. Rao, M. Lindorfer, A. Legout, and D. Choffnes. ReCon: Revealing and Controlling Privacy Leaks in Mobile Network Traffic. In *In Proc. ACM MobiSys*, 2016.

[70] I. Reyes, P. Wijesekera, A. Razaghpanah, J. Reardon, N. Vallina-Rodriguez, S. Egelman, and S. Kreibich. "Is Our Children's Apps Learning?" Automatically Detecting COPPA Violations. In *IEEE ConPro*, 2017.

[71] N. Sadeh, A. Acquisti, T. D Breaux, L. Cranor, A. M. McDonald, J. R. Reidenberg, N. A. Smith, F. Liu, N. C. Russell, F. Schaub, et al. The Usable Privacy Policy Project. Technical report, Technical Report, CMU-ISR-13-119, Carnegie Mellon University, 2013.

[72] Samet Privacy, LLC. Official membership page. https://www.kidsafeseal.com/certifiedproducts/kidzinmind_app.html. Accessed: September 29, 2017.

[73] Samet Privacy, LLC. Official membership page. https://www.kidsafeseal.com/certifiedproducts/familytime_app.html. Accessed: September 29, 2017.

[74] Samet Privacy, LLC. Member list. https://www.kidsafeseal.com/certifiedproducts.html, 2011. Accessed: November 30, 2017.

[75] E.J. Schwartz, T. Avgerinos, and D. Brumley. All You Ever Wanted to Know About Dynamic Taint Analysis and Forward Symbolic Execution (but Might Have Been Afraid to Ask). In *Proc. of the IEEE Symposium on Security and Privacy (SP)*, Oakland '10, 2010.

[76] Sirsi Corporation. Legal & privacy terms. http://www.sirsidynix.com/privacy, April 23 2004. Accessed: September 29, 2017.

[77] Y. Song and U. Hengartner. PrivacyGuard: A VPN-based Platform to Detect Information Leakage on Android Devices. In *Proc. of ACM SPSM*, 2015.

[78] Tapjoy, Inc. Publishers terms of service. https://home.tapjoy.com/legal/publishers-terms-service/, February 16 2016. Accessed: September 29, 2017.

[79] Upsight. COPPA. https://help.upsight.com/api-sdk-reference/integration-checklist/#coppa, 2017. Accessed: November 30, 2017.

[80] U.S. Court of Appeals, Ninth Circuit. Oracle USA, Inc. v. Rimini Street, Inc. https://www.eff.org/document/oracle-v-rimini-ninth-circuit-opinion. Accessed: March 24, 2018.

[81] U.S. Federal Trade Commission. Coppa safe harbor program. https://www.ftc.gov/safe-harbor-program. Accessed: September 28, 2017.

[82] U.S. Federal Trade Commission. FTC Approves Modifications to TRUSTe's COPPA Safe Harbor Program. https://www.ftc.gov/news-events/press-releases/2017/07/ftc-approves-modifications-trustes-coppa-safe-harbor-program. Accessed: September 28, 2017.

[83] U.S. Federal Trade Commission. Mobile apps for kids: Disclosures still not making the grade. https://www.ftc.gov/sites/default/files/documents/reports/mobile-apps-kids-disclosures-still-not-making-grade/121210mobilekidsappreport.pdf, December 2012.

[84] U.S. Federal Trade Commission. Children's online privacy protection rule: A six-step compliance plan for your business. https://www.ftc.gov/tips-advice/business-center/guidance/childrens-online-privacy-protection-rule-six-step-compliance, June 2017. Accessed: November 30, 2017.

[85] E. van der Walt and J. Eloff. Protecting Minors on Social Media Platforms-A Big Data Science Experiment. *Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam*, page 15, 2015.

[86] M. Van Kleek, I. Liccardi, R. Binns, J. Zhao, D.J. Weitzner, and N. Shadbolt. Better the Devil you Know: Exposing the Data Sharing Practices of Smartphone Apps. In *Proc. of ACM CHI*, 2017.

[87] WiGLE. Wigle: Wirless network mapping. https://wigle.net/. Accessed: September 29, 2017.

[88] P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. Wagner, and K. Beznosov. Android Permissions Remystified: A Field Study on Contextual Integrity. In *Proc. of USENIX Security*, 2015.

[89] P. Wijesekera, A. Baokar, L. Tsai, J. Reardon, S. Egelman, D. Wagner, and K. Beznosov. The Feasability of Dynamically Granted Permissions:
Aligning Mobile Privacy with User Preferences. In *Proc. of IEEE Symposium on Security and Privacy (SP)*, Oakland '17, 2017.

[90] B. Yankson, F. Iqbal, and P.C.K. Hung. Privacy preservation framework for smart connected toys. In *Computing in Smart Toys*, pages 149–164. Springer, 2017.

[91] S. Yong, D. Lindskog, R. Ruhl, and P. Zavarsky. Risk Mitigation Strategies for Mobile Wi-Fi Robot Toys from Online Pedophiles. In *Proc. of IEEE SocialCom*, pages 1220–1223. IEEE, 2011.

[92] S. Zimmeck, Z. Wang, L. Zou, R. Iyengar, B. Liu, F. Schaub, S. Wilson, N. Sadeh, S. M. Bellovin, and J. Reidenberg. Automated Analysis of Privacy Requirements for Mobile Apps. In *Proc. of NDSS Symposium*, 2017.

[93] S. Zimmeck, Z. Wang, L. Zou, R. Iyengar, B. Liu, F. Schaub, S. Wilson, N. Sadeh, S.M. Bellovin, and J.R. Reidenberg. Automated Analysis of Privacy Requirements for Mobile Apps. In *Proc. of NDSS Symposium*, 2017.

A SDK Terms of Use

Crashlytics: *Developer further agrees it will not integrate the Software into any Application or Beta Application (i) with end users who Developer has actual knowledge are under the age of 13, or (ii) that may be deemed to be a "Web site or online service directed to children" as defined under the Children's Online Privacy Protection Act of 1998 ("COPPA") and the regulations promulgated thereunder.* [39]

MoPub: *Supply Partners who sign up using this website may not provide MoPub with data from end users under age 13. Supply Partners must not register for MoPub's services using this website if any of their apps are either: (1) directed to children under age 13 (even if children are not the app's primary audience), or (2) collect in-*

formation from children that Supply Partners know are under age 13. [61]

Tapjoy: *Publisher represents, warrants, and covenants that (i) its Application(s) are not and shall not during the Term be directed to users under 13 years of age; (ii) Publisher does not as of the date Publisher creates a Publisher Account, and will not during the Term, collect, use, or disclose personal information from any end user known to Publisher to be a child under 13.* [78]

Branch: *Except as expressly permitted under these Terms, you will not, and will not permit anyone else to:...(g) use the Services in connection with any Apps or websites that are directed to children under 13.* [12]

Supersonic / ironSource: *The Services are not directed to children under the age of 13 and children under the age of 13 should not use any portion of the Services. ironSource also does not knowingly collect or maintain personal information collected online from children under the age of 13, to the extent prohibited by the Children's Online Privacy Protection Act. Nor do we knowingly create profile segments of children under 13 years of age.* [47]

Heyzap: *You must not include functionality in your application and/or website that requests or collects personal identification information from users who You know or have reason to know may be under the age of 13.* [44]

Amplitude: *Amplitude.com and the Services are not intended for anyone under the age of 13. Amplitude does not knowingly collect information from anyone under the age of 13. No one under the age of 13 may access or use the Services to provide Amplitude with personally identifiable information.* [4]

Appboy: *You shall not knowingly collect Personal Information of children under the age of 13. If the Customer Application is developed, marketed, advertised or directed to children under 13, or if the Customer Application collects Personal Information of children under 13, You represent that it has parental consent to collect such Personal Information of children under 13.* [5]

Appnext: *Company represents and warrants that its Property...does not contain any Objectionable Content, and is not directed to or primarily appeals to children under the age of 13...* [6]

Inneractive: *the App and its Content:...is not an online service directed to children ("Child-Directed") under the age of 13 or 16 as determined by the applicable data protection law (*e.g.,*. EU Directive 2016/679; Children Online Privacy Protection Act – "COPPA"), and to the extent that the App is Child-Directed, you will notify Inneractive in writing with each Ad request that such request originated from a child under the age of 13 or 16 (as applicable) and will not transmit any "Personal Information" (as defined under COPPA) or personal data as defined under the applicable EU Directive about or relating to an individual under the age of 13 or 16 (as applicable) to Inneractive.* [46]

B Safe Harbor Violations

- **Finny (com.app.psvbalance, v4.1)**: Certified by PRIVO [31]. Location data is sent to `dev.appboy.com`, which is a third party not disclosed in Finny's privacy policy.
- **KidzInMind (com.buongiorno.kim, v6.2.1e)**: Certified by kidSAFE [72]. Email address is sent to `client-api.newton.pm`. The privacy policy states that they "do not knowingly collect or solicit information from children and minors" [13], despite the fact that they are collecting PII without verifiable parental consent from a child-directed app.
- **ClassDojo (com.classdojo.android, v4.6.3)**: Certified by iKeepSafe [19]. Location data is sent to `api.amplitude.com`, whose privacy policy prevents it from being used in child-directed apps [4].
- **Rail Rush (com.miniclip.railrush, v1.9.12)**: Certified by CARU [18]. Location data is sent to `ads.aerserv.com`, `analytics.mopub.com`, and `ads.mopub.com`, the latter of which is sent over port 80 (unencrypted). MoPub's terms of service prohibits its inclusion in child-directed apps [61]. Email address is transmitted to `api.fuelpowered.com`, which does not "knowingly collect personal data from children under the age of 13" [32]. The game's privacy policy says that it does "not disclose personal data of users under 13 years of age to any third parties" [59].
- **NFL Draft (com.nfl.mobile.draft, v5.29.36)**: Certified by TRUSTe [62]. The privacy policy states that "the TRUSTe program covers only information that is collected through our Services that link to this Privacy Policy" (the app links to this privacy policy, suggesting it is covered by the TRUSTe program) [62]. Location data is sent to `placebubble.gimbal.com`. The privacy policy further states that this information may be used for "geographically relevant advertising" [62], which is likely a prohibited practice.
- **NFL Emojis (com.swyft.nfl, v1.1)**: Certified by TRUSTe [62]. Email address is sent to

control.kochava.com, which is not a third party named in the privacy policy [62].

– **FamilyTime Parental Controls & Time Management App (io.familytime.dashboard, v1.3.3.142)**: Certified by kidSAFE [73]. Email address is sent to admin.appnext.com, an advertising SDK, whose terms of service state that an app using it "is not directed to or primarily appeals to children under the age of 13" and that the app "shall not provide to Appnext any data regarding children under the age of 13" [6]. Appnext is not listed in the FamilyTime privacy policy [30].
– **BookMyne (sirsidynix.bookmyne, v4.1.1)**: Certified by TRUSTe [76]. Location data is sent to bookmyne.bc.sirsidynix.net, which is not disclosed in their privacy policy [76].