LinkDroid: Reducing Unregulated Aggregation of App Usage Behaviors

Huan Feng, Kassem Fawaz, and Kang G. Shin
Department of Electrical Engineering and Computer Science
The University of Michigan
{*huanfeng, kmfawaz, kgshin*}*@umich.edu*

Abstract

Usage behaviors of different smartphone apps capture different views of an individual's life, and are largely independent of each other. However, in the current mobile app ecosystem, a curious party can covertly link and aggregate usage behaviors of the same user across different apps. We refer to this as *unregulated aggregation* of app-usage behaviors. In this paper, we present a fresh perspective of unregulated aggregation, focusing on monitoring, characterizing and reducing the underlying linkability across apps. The cornerstone of our study is the *Dynamic Linkability Graph* (DLG) which tracks app-level linkability during runtime. We observed how DLG evolves on real-world users and identified real-world evidence of apps abusing IPCs and OS-level identifying information to establish linkability. Based on these observations, we propose a linkability-aware extension to current mobile operating systems, called `LinkDroid`, which provides runtime monitoring and mediation of linkability across different apps. `LinkDroid` is a client-side solution and compatible with the existing smartphone ecosystem. It helps end-users "sense" this emerging threat and provides them intuitive opt-out options.

1 Introduction

Mobile users run apps for various purposes, and exhibit very different or even unrelated behaviors in running different apps. For example, a user may expose his chatting history to WhatsApp, mobility traces to Maps, and political interests to CNN. Information about a single user, therefore, is scattered across different apps and each app acquires only a partial view of the user. Ideally, these views should remain as 'isolated islands of information' confined within each of the different apps. In practice, however, once the users' behavioral information is at the hands of the apps, it may be shared or leaked in an arbitrary way without the users' control or consent. This makes it possible for a curious adversary to aggregate usage behaviors of the same user across multiple apps without his knowledge and consent, which we refer to as *unregulated aggregation* of app-usage behaviors.

In the current mobile ecosystem, many parties are interested in conducting unregulated aggregation, including:

- *Advertising Agencies* embed ad libraries in different apps, establishing an explicit channel of cross-app usage aggregation. For example, Grindr is a geosocial app geared towards gay users, and BabyBump is a social network for expecting parents. Both apps include the same advertising library, MoPub, which can aggregate their information and recommend related ads, such as on gay parenting books. However, users may not want this type of unsolicited aggregation, especially across sensitive aspects of their lives.

- *Surveillance Agencies* monitor all aspects of the population for various precautionary purposes, some of which may cross the 'red line' of individuals' privacy. It has been widely publicized that NSA and GCHQ are conducting public surveillance by aggregating information leaked via mobile apps, including popular ones such as Angry Birds [3]. A recent study [26] shows that a similar adversary is able to attribute up to 50% of the mobile traffic to the "monitored" users, and extract detailed personal interests, such as political views and sexual orientations.

- *IT Companies* in the mobile industry frequently acquire other app companies, harvesting vast user base and data. Yahoo alone acquired more than 10 mobile app companies in 2013, with Facebook and Google following closely behind [1]. These acquisitions allow an IT company to link and aggregate behaviors of the same user from multiple apps without the user's consent. Moreover, if the acquiring com-

pany (such as Facebook) already knows the users' real identities, usage behaviors of all the apps it acquires become identifiable.

These scenarios of unregulated aggregation are realistic, financially motivated, and are only becoming more prevalent in the foreseeable future. In spite of this grave privacy threat, the process of unregulated aggregation is unobservable and works as a black box — no one knows what information has actually been aggregated and what really happens in the cloud. Users, therefore, are largely unaware of this threat and have no opt-out options. Existing proposals disallow apps from collecting user behaviors and shift part of the app logic (e.g., personalization) to the mobile OS or trusted cloud providers [7, 17]. This, albeit effective, is against the incentive of app developers and requires construction of a new ecosystem. Therefore, there is an urgent need for a practical solution that is compatible with the existing mobile ecosystem.

In this paper, we propose a new way of addressing the unregulated aggregation problem by monitoring, characterizing and reducing the underlying linkability across apps. Two apps are *linkable* if they can associate their usage behaviors of the same user. This linkability is the prerequisite of conducting unregulated aggregation and represents an upper-bound of the potential threat. Researchers studied linkability under domain-specific scenarios, such as on movie reviews [19] and social networks [16]. In contrast, we focus on the linkability that is ubiquitous in the mobile ecosystem and introduced by domain-independent factors, such as device IDs, account numbers, location and inter-app communications. Specifically, we model mobile apps on the same device as a *Dynamic Linkability Graph* (DLG) which monitors apps' access to OS-level identifying information and cross-app communication channels. DLG quantifies the potential threat of unregulated aggregation and allows us to monitor the linkability across apps during runtime.

We implemented DLG as an Android extension and observed how it evolved on 13 users during a period of 47 days. The results reveal an alarming view of the app-level linkability in the wild. Two random apps (installed by the same user) are linkable with a probability of 0.81. Specifically, 86% of the apps a user installed are directly linkable to the Facebook app, namely, his real identity. In particular, we found that apps frequently abuse OS-level information and inter-process communication (IPC) channels in unexpected ways, establishing the linkability that is unrelated to app functionalities. For example, we found that many of the apps requesting account information collect all of the user's accounts even when they only need one to function correctly. We also noticed that some advertising agencies, such as Admob and Facebook, use IPCs to share user identifiers with other apps, completely bypassing system permissions and controls. Furthermore, we identified cases when different apps write and read the same persistent file in shared storage to exchange user identifiers. The end-users should be promptly warned about these unexpected behaviors to reduce unnecessary linkability.

Based on the above observations, we propose `LinkDroid`, a linkability-aware extension to Android which provides runtime monitoring and mediation of the linkability across apps. `LinkDroid` introduces a new dimension to privacy protection on smartphones. Instead of checking whether some app behavior poses direct privacy threat, `LinkDroid` warns users about how it implicitly affects the linkability across apps. Practicality is a main driver for the design of `LinkDroid`. It extends the widely-deployed (both runtime and install-time) permission model on the mobile OS that end-users are already familiar with. Specifically, `LinkDroid` provides the following privacy-enhancing features:

- **Install-Time Obfuscation:** `LinkDroid` obfuscates device-specific identifiers that have no influence on most app functionalities, such as IMEI, Android ID, etc. We perform this during install-time to maintain the consistency of these identifiers within each app.
- **Runtime Linkability Monitoring:** When an app tries to perform a certain action that introduces additional linkability, users will receive a just-in-time prompt and an intuitive risk indicator. Users can then exercise runtime access control and choose any of the opt-out options provided by `LinkDroid`.
- **Unlinkable Mode:** The user can start an app in unlinkable mode. This will create a new instance of the app which is unlinkable with other apps. All actions that may establish a direct association with other apps will be denied by default. This way, users can enjoy finer-grained privacy protection, unlinking only a set of app sessions.

We evaluated `LinkDroid` on the same set of 13 users as in our measurement and found that `LinkDroid` reduces the cross-app linkability substantially with little loss of app performance. The probability of two random apps being linkable is reduced from 0.81 to 0.21, and the percentage of apps that are directly linkable to Facebook drops from 86% to 18%. On average, a user only needs to handle 1.06 prompts per day in the 47-day experiments and the performance overhead is marginal.

This paper makes the following contributions:

1. Introduction of a novel perspective of defending against unregulated aggregation by addressing the underlying linkability across apps (Section 2).

2. Proposal of the Dynamic Linkability Graph (DLG) which enables runtime monitoring of cross-app linkability (Section 3).

3. Identification of real-world evidence of how apps abuse IPCs and OS-level information to establish linkability across apps (Section 4).

4. Addition of a new dimension to access control based on the runtime linkability, and development of a practical countermeasure, `LinkDroid`, to defend against unregulated aggregation (Section 5).

2 Privacy Threats: A New Perspective

In this section, we will first introduce our threat model of unregulated aggregation and then propose a novel perspective of addressing it by monitoring, characterizing and reducing the linkability across apps. We will also summarize the explicit/implicit sources of linkability in the current mobile app ecosystem.

2.1 Threat Model

In this paper, we target unregulated aggregation across app-usage behaviors, i.e., when an adversary aggregates usage behaviors across multiple functionally-independent apps without users' knowledge or consent. In our threat model, an adversary can be any party that collects information from multiple apps or controls multiple apps, such as a widely-adopted advertising agency, an IT company in charge of multiple authentic apps, or a set of malicious colluding apps. We assume the mobile operating system and network operators are trustworthy and will not collude with the adversary.

2.2 Linkability: A New Perspective

There are many parties interested in conducting unregulated aggregation across apps. In practice, however, this process is unobservable and works as a black box — no one knows what information an adversary has collected and whether it has been aggregated in the cloud. Existing studies propose to disable mobile apps from collecting usage behaviors and shift part of the app logic to trusted cloud providers or mobile OS [7, 17]. These solutions, albeit effective, require building a new ecosystem and greatly restrict functionalities of the apps. Here, we address unregulated aggregation from a very different angle by monitoring, characterizing and reducing the underlying linkability across mobile apps. Two apps are *linkable* if they can associate usage behaviors of the same user. This linkability is the prerequisite of conducting unregulated aggregation, and represents an "upper-bound" of the potential threat. In the current mobile app ecosystem, there are various sources of linkability that an adversary can exploit. Researchers have studied linkability under several domain-specific scenarios, such as movie reviews [19] and social networks [16]. Here, we focus on the linkability that is ubiquitous and domain-independent. Specifically, we group its contributing sources into the following two fundamental categories.

Type	*2013-3*	*2013-10*	*2014-8*	*2015-1*
Android ID	80%	84%	87%	91%
IMEI	61%	64%	65%	68%
MAC	28%	42%	51%	55%
Account	24%	29%	32%	35%
Contacts	21%	26%	33%	37%

Table 1: Apps are increasingly interested in requesting persistent and consistent identifying information during the past few years.

OS-Level Information The mobile OS provides apps ubiquitous access to various system information, many of which can be used as consistent user identifiers across apps. These identifiers can be *device-specific*, such as MAC address and IMEI, *user-specific*, such as phone number or account number, or *context-based*, such as location or IP clusters. We conducted a longitudinal measurement study from March 2013 to January 2015, on the top 100 free Android apps in each category. We excluded the apps that are rarely downloaded, and considered only those with more than 1 million downloads. We found that apps are getting increasingly interested in requesting persistent and consistent identifying information, as shown in Table 1. By January 2015, 96% of top free apps request both the Internet access and at least one persistent identifying information. These identifying vectors, either explicit or implicit, allow two apps to link their knowledge of the same user at a remote side without even trying to bypass on-device isolation of the mobile OS.

Inter-Process Communications The mobile OS provides explicit Inter-Process Communication (IPC) channels, allowing apps to communicate with each other and perform certain tasks, such as export a location from Browser and open it with Maps. Since there is no existing control on IPC, colluding apps can exchange identifying information of the user and establish linkability covertly, without the user's knowledge. They can even synchronize and agree on a randomly-generated sequence as a custom user identifier, without accessing any system resource or permission. This problem gets more complex since apps can also conduct IPC implicitly by reading and writing shared persistent storage (SD card

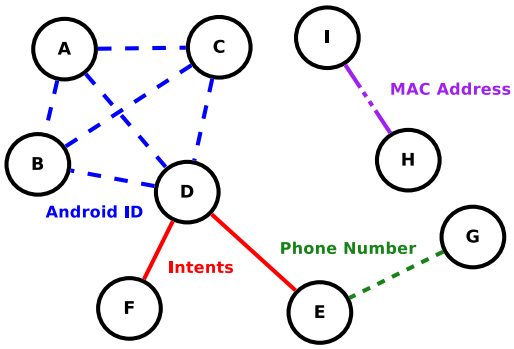

Figure 1: An illustrative example of DLG. Edges of different types represent linkability introduced by different sources.

and databases). As we will show in Section 4, these exploitations are not hypothetical and have already been utilized by real-world apps.

3 Dynamic Linkability Graph

The cornerstone of our work is the Dynamic Linkability Graph (DLG). It enables us to monitor app-level linkability during runtime and quantify the linkability introduced by different contributing sources. In what follows, we will elaborate on the definition of DLG, the linkability sources it considers, and describe how it can be implemented as an extension of Android.

3.1 Basic Concepts

We model linkability across different apps on the same device as an undirected graph, which is called the *Dynamic Linkability Graph* (DLG). Nodes in DLG represent apps and edges represent linkability introduced by different contributing sources. DLG monitors the linkability during runtime by tracking the apps' access to various OS-level information and IPC channels. An edge exists between two apps if they accessed the same identifying information or engaged in an IPC. Fig. 15 presents an illustrative example of DLG.

DLG presents a comprehensive view of the linkability across all installed apps. An individual adversary, however, may only observe a subgraph of the DLG. For example, an advertising agency only controls those apps (nodes) that incorporate the same advertising library; an IT corporate only controls those apps (nodes) it has already acquired. In the rest of the paper, we focus on the generalized case (the entire DLG) instead of considering each adversary individually (subgraphs of DLG).

3.2 Definitions and Metrics

Linkable Two apps a and b are *linkable* if there is a path between them. In Fig. 15, app A and F are linkable, app A and H are not linkable.

Gap is defined as the number of nodes (excluding the end nodes) on the shortest path between two linkable apps a and b. It represents how many additional apps an adversary needs to control in order to link information across a and b. For example, in Fig. 15, $gap_{A,D} = 0$, $gap_{A,E} = 1$, $gap_{A,G} = 2$.

Linking Ratio (LR) of an app is defined as the number of apps it is linkable to, divided by the number of all installed apps. *LR* ranges from 0 to 1 and characterizes to what extent an app is linkable to others. In DLG, *LR* equals to the size of the *Largest Connected Component* (LCC) this app resides in, excluding itself, divided by the size of the entire graph, also excluding itself:

$$LR_a = \frac{size(LCC_a) - 1}{size(DLG) - 1}$$

Linking Effort (LE) of an app is defined as the *Linking Effort* (LE) of an app as the average *gap* between it and all the apps it is linkable to. LE_a characterizes the difficulty in establishing linkability with a. $LE_a = 0$ means that to link information from app a and any random app it is linkable to, an adversary does not need additional information from a third app.

$$LE_a = \sum_{\substack{b \in LCC_a \\ b \neq a}} \frac{gap_{a,b}}{size(LCC_a) - 1}$$

LR and *LE* describe two orthogonal views of the DLG. In general, *LR* represents the quantity of links, describing the percentage of all installed apps that are linkable to a certain app, whereas *LE* characterizes the quality of links, describing the average amount of effort an adversary needs to make to link a certain app with other apps. In Fig. 15, $LR_A = 6/8$, $LR_H = 1/8$; $LE_A = \frac{0+0+0+1+1+2}{7-1} = 4/6$, $LE_H = 0$.

GLR and GLE Both *LR* and *LE* are defined for a single app, and we also need two similar definitions for the entire graph. So, we introduce *Global Linking Ratio* (GLR) and *Global Linking Effort* (GLE). *GLR* represents the probability of two randomly selected apps being linkable, while *GLE* represents the number of apps an adversary needs to control to link two random apps.

$$GLR = \sum_a \frac{LR_a}{size(DLG)}$$

$$GLE = \frac{1}{\sum_a size(LCC_a) - 1} \sum_b \sum_{\substack{c \in LCC_b \\ c \neq b}} gap_{b,c}$$

In graph theory, *GLE* is also known as the *Characteristic Path Length* (CPL) of a graph, which is widely used in Social Network Analysis (SNA) to characterize whether the network is easily negotiable or not.

3.3 Sources of Linkability

DLG maintains a dynamic view of app-level linkability by monitoring runtime behaviors of the apps. Specifically, it keeps track of apps' access to *device-specific* identifiers (IMEI, Android ID, MAC), *user-specific* identifiers (Phone Number, Accounts, Subscriber ID, ICC Serial Number), and *context-based* information (IP, Nearby APs, Location). It also monitors explicit IPC channels (Intent, Service Binding) and implicit IPC channel (Indirect RW, i.e., reading and writing the same file or database). This is not an exhaustive list but covers most standard and widely-used aggregating channels. Table 2 presents a list of all the contributing sources we consider and the details of each source will be elaborated in Section 3.4.

The criterion of two apps being linkable differs depending on the linkability source. For consistent identifiers that are obviously unique — Android ID, IMEI, Phone Number, MAC, Subscriber ID, Account, ICC Serial Number — two apps are linkable if they both accessed the same type of identifier. For pair-wise IPCs — intents, service bindings, and indirect RW — the two communicating parties involved are linkable. For implicit and fuzzy information, such as location, nearby APs, and IP, there are well-known ways to establish linkability as well. User-specific location clusters (Points of Interests, or PoIs) is already known to be able to uniquely identify a user [11, 15, 29]. Therefore, an adversary can link different apps by checking whether the location information they collected reveal the same PoIs. Here, the PoIs are extracted using a lightweight algorithm as used in [5, 10]. We select the top 2 PoIs as the linking standard, which typically correspond to home and work addresses. Similarly, the consistency and persistence of a user's PoIs are also reflected on its AP clusters and frequently-used IP addresses. This property allows us to establish linkability across apps using these fuzzy contextual information.

3.4 DLG: A Mobile OS Extension

DLG gives us the capability to construct cross-app linkability from runtime behaviors of the apps. Here, we introduce how it can be implemented as an extension to current mobile operating systems, using Android as an illustrative example. We also considered other implementation options, such as user-level interception (Aurasium [28]) or dynamic OS instrumentation (Xposed Framework [27]). The former is insecure since the extension resides in the attacker's address space and the latter is not comprehensive because it cannot handle the native code of an app. However, the developer can always implement a useful subset of DLG using one of these more deployable techniques.

Category	*Type*	*Source*
OS-level Info.	Device	IMEI Android ID MAC
	Personal	Phone # Account Subscriber ID ICC Serial #
	Contextual	IP Nearby APs Location (PoIs)
IPC Channel	Explicit	Intent Service Binding
	Implicit	Indirect RW

Table 2: DLG considers the linkability introduced by 10 types of OS-level information and 3 IPC channels.

Android Basics Android is a Linux-based mobile OS developed by Google. By default, each app is assigned a different Linux uid and lives in its own sandbox. Inter-Process Communications (IPCs) are provided across different sandboxes, based on the Binder protocol which is inherently a lightweight RPC (Remote Procedure Call) mechanism. There are four different types of components in an Android app: Activity, Service, Content Provider, and Broadcast Receiver. Each component represents a different way to interact with the underlying system: Activity corresponds to a single screen supporting user interactions; Service runs in the background to perform long-running operations and processing; Content Provider is responsible for managing and querying of persistent data such as database; and Broadcast Receiver listens to system-wide broadcasts and filters those it is interested in. Next, we describe how we instrument the Android framework to monitor app's interactions with the system and each other via these components.

Implementation Details In order to construct a DLG in Android, we need to track apps' access to various OS-

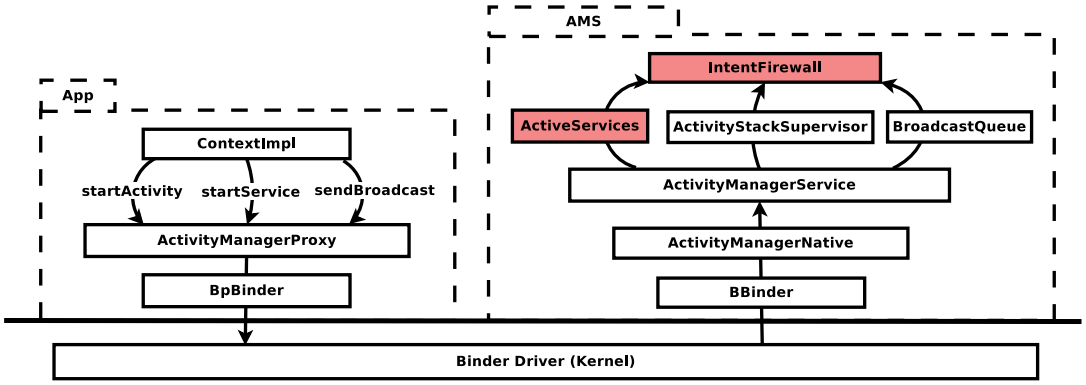

Figure 3: We extend the centralized intent filter in Android (`com.android.server.firewall.IntentFirewall`) to intercept all the intents across apps.

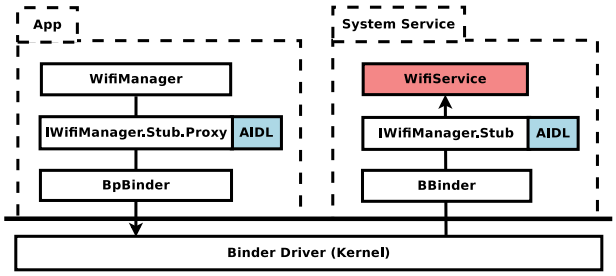

Figure 2: We instrument system services (red shaded region) to record which app accessed which identifier using Wi-Fi service as an example.

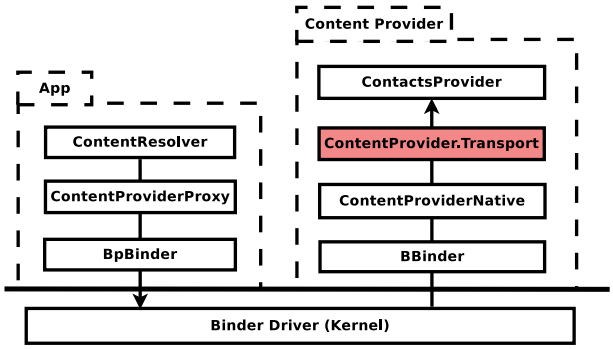

Figure 4: We instrument Content Provider (shaded region) to record which app accessed which database with what parameters.

level information as well as IPCs between apps. Next, we describe how we achieve this by instrumenting different components of the Android framework.

Apps access most identifying information, such as IMEI and MAC, by interacting with different system services. These system services are parts of the Android framework and have clear interfaces defined in AIDL (Android Interface Definition Language). By instrumenting the public functions in each service that return persistent identifiers, we can have a timestamped record of which app accessed what type of identifying information via which service. Fig. 2 gives a detailed view of where to instrument using the Wi-Fi service as an example.

On the other hand, apps access some identifying information, such as Android ID, by querying system content providers. Android framework has a universal choke point for all access to remote content providers — the server-side stub class `ContentProvider.Transport`. By instrumenting this class, we know which database (uri) an app is accessing and with what parameters and actions. Fig. 4 illustrates how an app accesses remote Content Provider and explains which part to modify in order to log the information we need.

Apps can launch IPCs explicitly, using Intents. Intent is an abstract description of an operation to be performed. It can either be sent to a specific target (app component), or broadcast to the entire system. Android has a centralized filter which enforces system-wide policies for all Intents. We extend this filter (`com.android.server.firewall.IntentFirewall`) to record and intercept all Intent communications across apps (see Fig. 3). In addition to Intents, Android also allows an app to communicate explicitly with another app by binding to one of the services it exports. Once the binding is established, the two apps can communicate under a client-server model. We instrument `com.android.server.am.ActiveServices` in the Activity Manager to monitor all the attempts to establish service bindings across apps.

Apps can also conduct IPCs implicitly by exploiting shared persistent storage. For example, two apps can write and read the same file in the SD card to exchange identifying information. Therefore, we need to monitor read and write access to persistent storage. External storage in Android are wrapped by a FUSE (Filesystem in

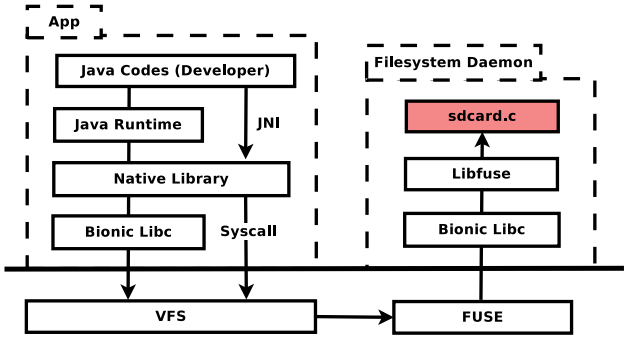

Figure 5: We customize the FUSE daemon under `/system/core/sdcard/sdcard.c` to intercept apps' access to shared external storage.

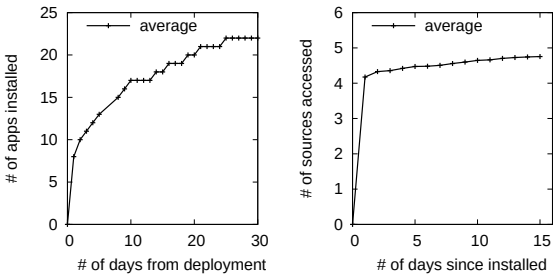

Figure 6: For an average user, more than 80% of the apps are installed in the first two weeks after deployment; each app accesses most of the linkability sources it's interested in during the first day of its installation.

Userspace) daemon which enables user-level permission control. By modifying this daemon, we can track which app reads or writes which files (see Fig. 5). This allows us to implement a Read-Write monitor which captures implicit communications via reading a file which has previously been written by another app. Besides external storage, our Read-Write monitor also considers similar indirect communications via system Content Providers.

We described how to monitor all formal ways an app can interact with system components (Services, Content Providers) and other apps (Intents, service bindings, and indirect RW). This methodology is fundamental and can be extended to cover other potential linkability sources (beyond our list) as long as a clear definition is given. By placing hooks at the aforementioned locations in the system framework, we get all the information needed to construct a DLG. For our measurement study, we simply log and upload these statistics to a remote server for analysis. In our countermeasure solutions, these are used locally to derive dynamic defense decisions.

4 Linkability in Real World

In this section, we study app-level linkability in the real world. We first present an overview of linkability, showing the current threats we're facing. Then, we go through the linkability sources and analyze to what extent each of the sources is contributing to the linkability. Finally, we shed light on how these sources can be or have been exploited for reasons unrelated to app functionalities. This paves the way for us to develop a practical countermeasure.

4.1 Deployment and Settings

We prototyped DLG on Cyanogenmod 11 (based on Android 4.4.1) and installed the extended OS on 7 Samsung Galaxy IV devices and 6 Nexus V devices. We recruited 13 participants from the students and staff in our institution, spanning over 8 different academic departments. Of the 13 participants, 6 of the participants are females and 7 are males. Before using our experimental devices, 7 of them were Android users and 6 were iPhone users. Participants are asked to operate their devices normally without any extra requirement. They are given the option to temporarily turn off our extension if they want more privacy when performing certain tasks. Logs are uploaded once per hour when the device is connected to Wi-Fi. We exclude built-in system apps (since the mobile OS is assumed to be benign in our threat model) and consider only third-party apps that are installed by the users themselves. Note that our study is limited in its size and the results may not generalize.

4.2 Data and Findings

We observed a total of 215 unique apps during a 47-day period for 13 users. On average, each user installed 26 apps and each app accessed 4.8 different linkability sources. We noticed that more than 80% of the apps are installed within the first two weeks after deployment, and apps would access most of the linkability sources they are interested in during the first day of their installation (see Fig. 6). This suggests that a relative short-term (a few weeks) measurement would be enough to capture a representative view of the problem.

Overview: Our measurement indicates an alarming view of the threat: two random apps are linkable with a probability of 0.81, and an adversary only needs to control 2.2 apps (0.2 additional app), on average, to link them. This means that an adversary in the current ecosystem can aggregate information from most apps without additional efforts (i.e., controlling a third app). Specifically, we found that 86% of the apps a user installed on his device are directly linkable to the Facebook app, namely, his real identity. This means almost all the activ-

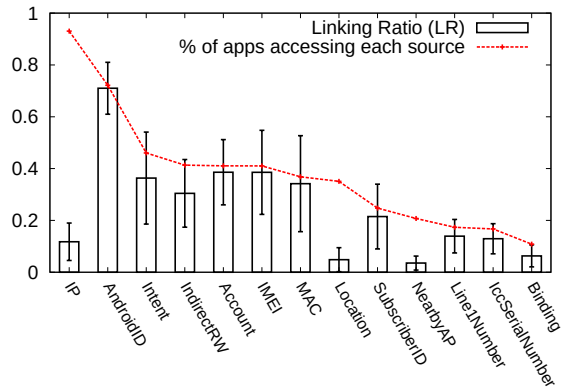

Figure 7: The percentage of apps accessing each source, and the linkability (LR) an app can get by exploiting each source.

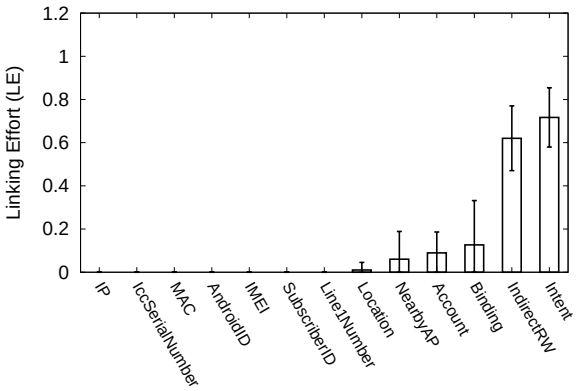

Figure 8: The (average) Linking Efforts (LE) of all the apps that are linkable due to a certain linkability source.

ities a user exhibited using mobile apps are identifiable, and can be linked to the real person.

Breakdown by Source: This vast linkability is contributed by various sources in the mobile ecosystem. Here, we report the percentage of apps accessing each source and the linkability (LR) an app can acquire by exploiting each source. The results are provided in Fig. 7. We observed that except for device identifiers, many other sources contributed to the linkability substantially. For example, an app can be linked to 39% of all installed apps (LR=0.39) using only account information, and 36% (LR=0.36) using only Intents. The linkability an app can get from a source is roughly equal to the percentage of apps that accessed that source, except for the case of contextual information: IP, Location and Nearby APs. This is because the contextual information an app collected does not always contain effectively identifying information. For example, Yelp is mostly used at infrequent locations to find nearby restaurants, but is rarely used at consistent PoIs, such as home or office. This renders location information useless in establishing linkability with Yelp.

The effort required to aggregate two apps also differs for different linkability sources, as shown in Fig. 8. Device identifiers have *LE=0*, meaning that any two apps accessing the same device identifier can be directly aggregated without requiring control of an additional third app. Linking apps using IPC channels, such as Intents and Indirect RW, requires the adversary to control an average of 0.6 additional app as the connecting nodes. This indicates that, from an adversary's perspective, exploiting consistent identifiers is easier than building pair-wise associations.

Breakdown by Category: We group the linkability sources into four categories — device, personal, contextual, and IPC — and study the linkability contributed by each category (see Table 3). As expected, device-specific information introduces substantial linkability and allows the adversary to conduct across-app aggregation effortlessly. Surprisingly, the other three categories of linkability sources also introduce considerable linkability. In particular, only using fuzzy contextual information, an adversary can link more than 40% of the installed apps to Facebook, the user's real identity. This suggests the naive solution of anonymizing device ids is not enough, and hence a comprehensive solution is needed to make a trade-off between app functionality and privacy.

Category	*GLR*	*GLE*	$LR_{Facebook}$
Device	0.52 (0.13)	0.03 (0.03)	0.68 (0.12)
Personal	0.30 (0.10)	0.30 (0.11)	0.54 (0.11)
Contextual	0.20 (0.13)	0.33 (0.20)	0.44 (0.25)
IPC	0.32 (0.13)	0.78 (0.06)	0.59 (0.15)

Table 3: Linkability contributed by different categories of sources.

4.3 Functional Analysis

Device identifiers (IMEI, Android ID, MAC) introduce vast amount of linkability. We manually went through 162 mobile apps that request these device-specific identifiers, but could rarely identify any explicit functionality that requires accessing the actual identifier. In fact, for the majority of these apps, their functionalities are device-independent, and therefore independent of device IDs. This indicates that device-specific identifier can be obfuscated across apps without noticeable loss of app functionality. The only requirement for device ID is that it should be unique to each device.

As to personal information (Account Number, Phone

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <long name="timestamp" value="1419049777098" />
    <long name="t2" value="1419049776889" />
    <string name="UTDID">VJT7MTV268gDACiZN6xEh8af</string>
    <string name="DID">356565055348652</string>
    <long name="S" value="1634341681" />
    <string name="SI">310260981039000</string>
    <string name="EI">356565055348652</string>
</map>
```

Figure 9: Real-world example of indirect RW: an app (fm.qingting.qradio) writes user identifiers to an xml file in SD card which was later read by three other apps. This file contains the IMEI (DID) and SubscriberID (SI) of the user.

Number, Installed Apps, etc.), we also observed many unexpected accesses that resulted in unnecessary linkability. We found that many apps that request account information collected all user accounts even when they only needed one to function correctly; many apps request access to phone number even when it is unrelated to their app functionalities. Since the legitimacy of a request depends both on the user's functional needs and the specific app context, end-users should be prompted about the access and make the final decision.

The linkability introduced by contextual information (Location, Nearby AP) also requires better regulation. Many apps request permission for precise location, but not all of them actually need it to function properly. In many scenarios, apps only require coarse-grained location information and shouldn't reveal any identifying points of interest (PoIs). Nearby AP information, which is only expected to be used by Wi-Fi tools/managing apps, is also abused for other purposes. We noticed that many apps frequently collect Nearby AP information to build an internal mapping between locations and access points (APs). For example, we found that even if we turn off all system location services, WeChat (an instant messaging app) can still infer the user's location only with Nearby AP information. To reduce the linkability introduced by these unexpected usages, the users should have finer-grained control on when and how the contextual information can be used.

Moreover, we found that IPC channels can be exploited in various ways to establish linkability across apps. Apps can establish linkabililty using Intents, sharing and aggregating app-specific information. For instance, we observed that WeChat receives Intents from three different apps right after their installations, reporting their existence on the same device. Apps can also establish linkability with each other via service binding. For example, both AdMob and Facebook allow an app to bind to its service and exchanging the user identifier, completely bypassing the system permissions and controls. Apps can also establish linkabililty through Indirect RW, by writing and reading the same persistent file. Fig. 9 shows a real-world example: an app (fm.qingting.qradio) writes user identifiers to an xml file in the SD card which was later read by three other apps. The end-user should be promptly warned about these unexpected communications across apps to reduce unnecessary linkability.

5 LinkDroid: A Practical Countermeasure

Based on our observation and findings on linkability across real-world apps, we propose a practical countermeasure, `LinkDroid`, on top of DLG. We first introduce the basic design principle of `LinkDroid` and its three major privacy-enhancing features: *install-time obfuscation*, *runtime linkability monitoring*, and *unlinkable mode support*. We then evaluate the effectiveness of `LinkDroid` with the same set of participants as in our measurement study.

5.1 Design Overview

`LinkDroid` is designed with practicality in mind. Numerous extensions, paradigms and ecosystems have been proposed for mobile privacy, but access control (runtime for iOS and install-time for Android) is the only deployed mechanism. `LinkDroid` adds a new dimension to access control on smartphone devices. Unlike existing approaches that check if some app behavior poses direct privacy threats, `LinkDroid` warns users about how it implicitly builds the linkability across apps. This helps users reduce unnecessary links introduced by abusing OS-level information and IPCs, which happens frequently in reality as our measurement study indicated.

As shown in Fig. 10, `LinkDroid` provides runtime monitoring and mediation of linkability by

- monitoring and intercepting app behaviors that may introduce linkability (including interactions with various system services, content providers, shared external storage and other apps);
- querying a standalone linkability service to get the user's decision regarding this app behavior;
- prompting the user about the potential risk if the user has not yet made a decision, getting his decision and updating the linkability graph (DLG).

We have already described in Section 3.4 how to instrument the Android framework to build the monitoring components (corresponding to boxes A, B, C, D in Fig. 10). In this section, we focus on how the linkability service operates.

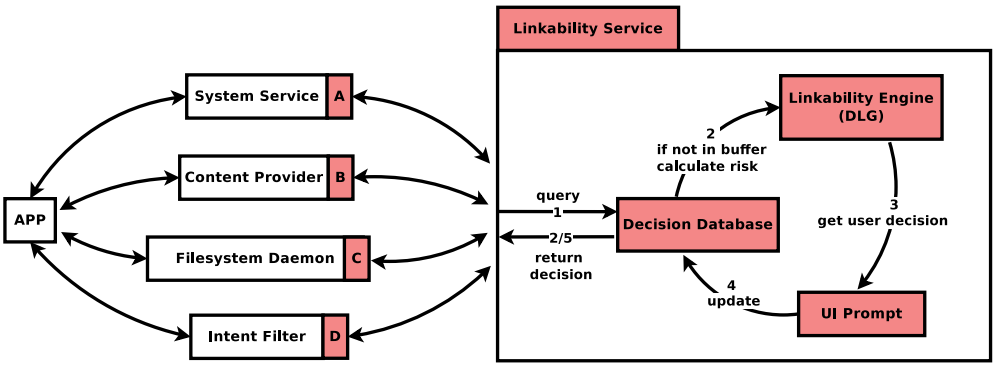

Figure 10: An overview of `LinkDroid`. Shaded areas (red) represent the parts we need to extend/add in Android. (We already explained how to extend A, B, C and D in Section 3.4.)

5.2 Install-Time Obfuscation

As mentioned earlier, app functionalities are largely independent of device identifiers. This allows us to obfuscate these identifiers and cut off many unnecessary edges in the DLG. In our case, the list of device identifiers includes IMEI, Android ID and MAC. Every time an app gets installed, the linkability service receives the app's uid and then generates a random mask code for it. The mask code together with the types of obfuscated device identifiers will be pushed into the decision database. This way, when an app a tries to fetch the device identifier of a certain type t, it will only get a hash of the real identifier salted with the app-specific mask code:

$$ID_t^a = hash(ID_t + mask_a).$$

Note that we do this at install-time instead of during each session because we still want to guarantee the relative consistency of the device identifiers within each app. Otherwise, it will let the app think the user is switching to a different device and trigger some security/verification mechanisms. The user can always cancel this default obfuscation in the privacy manager (Fig. 12) if he finds it necessary to reveal real device identifiers to certain apps.

5.3 Runtime Linkability Monitoring

Except for device-specific identifiers, obfuscating other sources of linkability is likely to interfere with the app functionalities. Whether there is a functional interference or not is highly user-specific and context-dependent. To make a useful trade-off, the user should be involved in this decision-making process. Here, `LinkDroid` provides just-in-time prompts before an edge creates in the DLG. Specifically, if the linkability service could not find an existing decision regarding some app behavior, it will issue the user a prompt, informing him: 1) what app behavior triggers the prompt; 2) what's the quantitative risk of allowing this behavior; and 3) what're the opt-out options. Fig. 11 gives an illustrative example of the UI of the prompt.

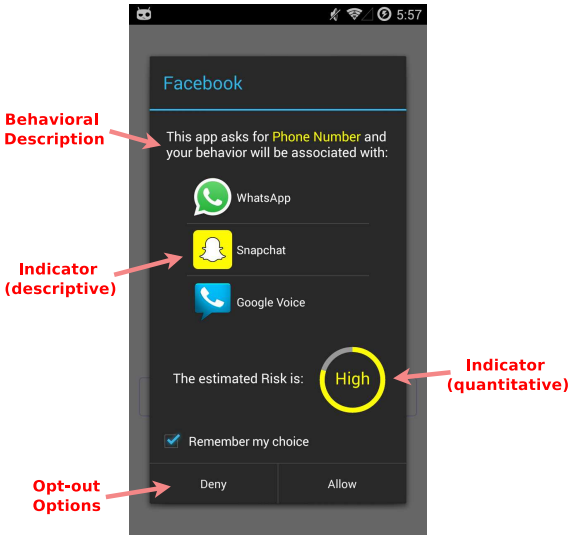

Figure 11: The UI prompt of `LinkDroid`'s runtime access control, consisting of a behavioral description, descriptive and quantitative risk indicators, and opt-out options.

Description of App Behavior Before the user can make a decision, he first needs to know what app behavior triggers the prompt. Basically, we report two types of description: access to OS-level information and cross-app communications. To help the user understand the situation, we use a high-level descriptive language instead of the exact technical terms. For example, when an app tries to access Subscriber ID or IccSerialNumber, we report that "App X asks for sim-card information." When an app tries to send Intents to other apps, we report "App X tries to share content with App Y". During our experiments with real users (introduced later in the evaluation), 11 out of the 13 participants find these descriptions clear

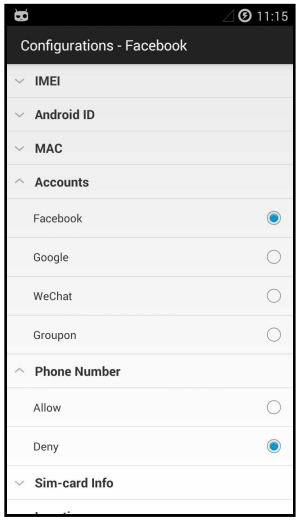

Figure 12: LinkDroid provides a centralized linkability manager. The user can review and modify all of his previous decisions regarding each app.

and informative.

Risk Indicator LinkDroid reports two types of risk indicators to users: one is descriptive and the other is quantitative. The descriptive indicator tells what apps will be directly linkable to an app if the user allows its current behavior. By 'directly linkable,' we mean without requiring a third app as the connecting nodes. The quantitative indicator, on the other hand, reflects the influence on the overall linkability of the running app, including those apps that are not directly linkable to it. Here, the overall linkability is reported as a combination of the linking ratio (LR) and linking effort (LE):

$$L_a = LR_a \times e^{-LE_a}.$$

The quantitative risk indicator is defined as ΔL_a. A user will be warned of a larger risk if the total number of linkable apps significantly increases, or the average linking effort decreases substantially. We transform the quantitative risk linearly into a scale of 4 and report the risk as Low, Medium, High, and Severe.

Opt-out Options In each prompt, the user has at least two options: Allow or Deny. If the user chooses Deny, LinkDroid will obfuscate the information this app tries to get or shut down the communication channel this app requests. For some types of identifying information, such as Accounts and Location, we provide finer-grained trade-offs. For Location, the user can select from zip-code level (1km) or city-level (10km) precision; for Accounts, the user can choose which specific account he wants to share instead of exposing all his accounts. LinkDroid also allows the user to set up a VPN (Virtual Private Network) service to anonymize network identifiers. When the user switches from a cellular network to Wi-Fi, LinkDroid will automatically initialize the VPN service to hide the user's public IP. This may incur additional energy consumption and latency (see Section 5.5). All choices made by the user will be stored in the decision database for future reuse. We provide a centralized privacy manager such that the user can review and change all previously made decisions (see Fig. 12).

5.4 Unlinkable Mode

Once a link is established in DLG, it cannot be removed. This is because once a piece of identifying information is accessed or a communication channel is established, it can never be revoked. However, the user may sometimes want to perform privacy-preserving tasks which have no interference with the links that have already been introduced. For example, when the user wants to write an anonymous post in Reddit, he doesn't want it to be linkable with any of his previous posts as well as other apps. LinkDroid provides an unlinkable mode to meet such a need. The user can start an app in unlinkable mode by pressing its icon for long in the app launcher. A new uid as well as isolated storage will be allocated to this unlinkable app instance. By default, access to all OS-level identifying information and inter-app communications will be denied. This way, LinkDroid creates the illusion that this app has just been installed on a brand-new device. The unlinkable mode allows LinkDroid to provide finer-grained (session-level) control, unlinking only a certain set of app sessions.

5.5 Evaluation

We evaluate LinkDroid in terms of its overheads in usability and performance, as well as its effectiveness in reducing linkability. We replay the traces of the 13 participants of our measurement study (see Section 4), prompt them about the privacy threat and ask for their decisions. This gives us the exact picture of the same set of users using LinkDroid during the same period of time. We instruct the user to make a decision in the most conservative way: the user will Deny a request only when he believes the prompted app behavior is not applicable to any useful scenario; otherwise, he will Accept the request.

The overhead of LinkDroid mainly comes from two parts: the usability burden of dealing with UI prompts and the performance degradation of querying the linkability service. Our experimental results show that, on average, each user was prompted only 1.06 times per day during the 47-day period. The performance degradation

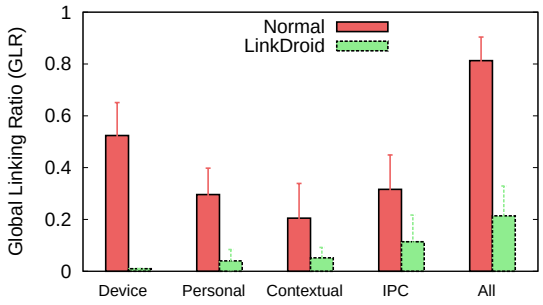

Figure 13: The Global Linking Ratio (GLR) of different categories of sources before and after using `LinkDroid`.

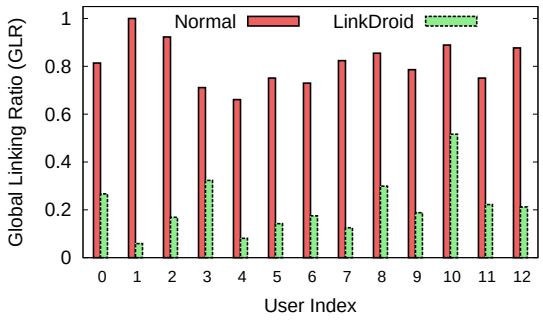

Figure 14: The Global Linking Ratio (GLR) of different users before and after using `LinkDroid`.

introduced by the linkability service is also marginal. It only occurs when apps access certain OS-level information or conduct cross-app IPCs. These sensitive operations happened rather infrequently — once every 12.7 seconds during our experiments. These results suggest that `LinkDroid` has limited impact on system performance and usability.

We found that after applying `LinkDroid`, the Global Linking Ratio (GLR) dropped from 81% to 21%. Fig. 13 shows the breakdown of linkability drop in different categories of sources. The majority of the remaining linkability comes from inter-app communications, most of which are genuine from the user's perspective. Not only fewer apps are linkable, `LinkDroid` also makes it harder for an adversary to aggregate information from two linkable apps. The Global Linking Effort (GLE) increases significantly after applying `LinkDroid`: from 0.22 to 0.68. Specifically, the percentage of apps that are directly linkable to Facebook dropped from 86% to 18%. Fig. 15 gives an illustrative example of how DLG changes after applying `LinkDroid`. We also noticed that that the effectiveness of `LinkDroid` differs across users, as shown in Fig. 14. In general, `LinkDroid` is more effective for the users who have diverse mobility patterns, are cautious about sharing information across apps and/or maintain

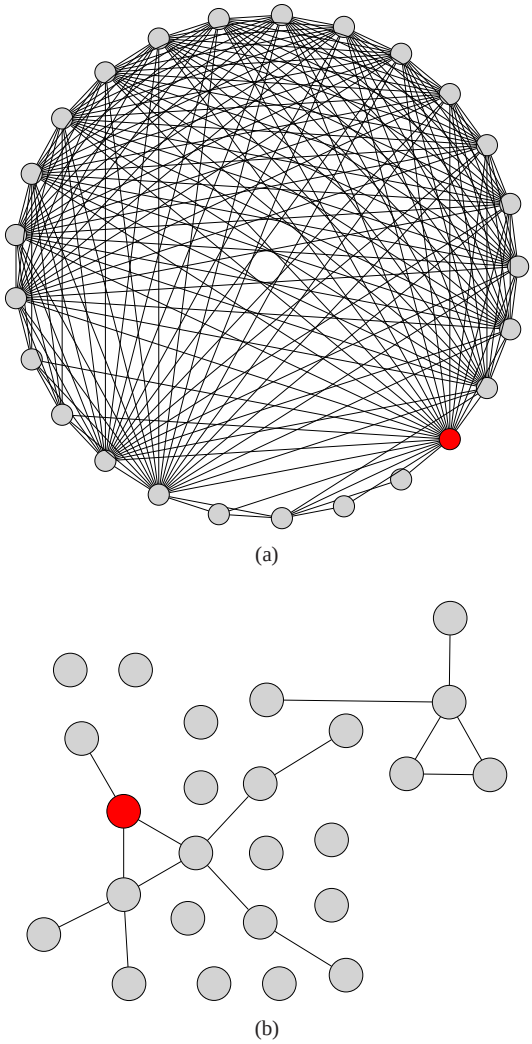

Figure 15: DLG of a representative user before (a) and after (b) applying `LinkDroid`. Red circle represents the Facebook app.

different accounts for different services.

`LinkDroid` takes VPN as a plug-in solution to obfuscate network identifiers. The potential drawback of using VPN is its influence on device energy consumption and network latency. We measured the device energy consumption of using VPN on a Samsung Galaxy 4 device, with Monsoon Power Monitor. Specifically, we tested two network-intensive workloads: online videos and browsing. We observed a 5% increase in energy consumption for the first workload, and no observable difference for the second. To measure the network latency, we measured the ping time (average of 10 trials) to Alexa Top 20 domains and found a 13% increase (17ms). These results indicate that the overhead of using VPN on smartphone device is noticeable but not significant. Seven of 13 participants in our evaluation were willing to use VPN

services to achieve better privacy.

We interviewed the 13 participants after the experiments. Questions are designed on a scale of 1 to 5 and a score of 4 or higher is regarded as "agree." Eleven of the participants find the UI prompt informative and clear and nine are willing to use `LinkDroid` on a daily basis to inform them about the risk and provide opt-out options. However, these responses might not be representative due to the limited size and diversity of the participants. We also noticed that users care a lot about the linkability of sensitive apps, such as Snapchat and Facebook. Some participants clearly state that they do not want any app to be associated with the Facebook app, except for very necessary occasions. This also supports the rationale behind the design of `LinkDroid`'s unlinkable mode.

6 Related Work

There have been other proposals [7, 17] which also address the privacy threats of information aggregation by mobile apps. They shift the responsibility of information personalization and aggregation from mobile apps to the mobile OS or trusted cloud providers, requiring re-development of mobile apps and extensive modifications on the entire mobile ecosystem. In contrast, `LinkDroid` is a client-side solution which is compatible with existing ecosystem — it focuses on characterizing the threat in current mobile ecosystem and making a practical trade-off, instead of proposing new computation (advertising) paradigm.

Existing studies investigated linkability under several domain-specific scenarios. Arvind *et al.* [19] showed that a user's profile in Netflix can be effectively linked to his in IMDB, using long-tailed (unpopular) movies. Sebastian *et al.* [16] described how to link the profiles of the same user in different social networks using friends topologies. This type of linkability is restricted to a small scope, and may only exist across different apps in the same domain. Here, we focus on the linkability that are domain-independent and ubiquitous to all apps, regardless of the type and semantics of each app.

The capability of advertising agency on conducting profiling and aggregation has been extensively studied [12, 23]. Various countermeasures have been proposed, such as enforcing finer-grained isolation between ad library and the app [21, 22], or adopting a privacy-preserving advertising paradigm [4]. However, unlike `LinkDroid`, they only consider a very specific and restricted scenario — advertising library — which involves few functional trade-offs. `LinkDroid`, instead, introduces a general linkability model, considers various sources of linkability and suits a diverse set of adversaries.

There have also been numerous studies on information access control on smartphone [6, 8, 9, 13, 14, 20, 24]. Many of these studies have already proposed to provide apps with fake identifiers and other types of sensitive information [13, 20, 27]. These studies focus on the explicit privacy concern of accessing and leaking sensitive user information, by malicious mobile apps or third-party libraries. Our work addresses information access control from a very different perspective, investigating the implicit linkability introduced by accessing various OS-level information and IPC channels.

Many modern browsers provide a private (incognito) mode. These are used to defend against local attackers, such as users sharing the same computer, from stealing cookies or browse history from each other [2]. This is inherently different from `LinkDroid`'s *unlinkable mode* which targets unregulated aggregation by remote attackers.

7 Discussion

In this paper, we proposed a new metric, *linkability*, to quantify the ability of different apps to link and aggregate their usage behaviors. This metric, albeit useful, is only a coarse upper-bound of the actual privacy threat, especially in the case of IPCs. Communication between two apps does not necessarily mean that they have conducted, or are capable of conducting, information aggregation. However, deciding on the actual intention of each IPC is by itself a difficult task. It requires an automatic and extensible way of conducting semantic introspection on IPCs, and is a challenging research problem on its own.

`LinkDroid` aims to reduce the linkability introduced covertly without the user's consent or knowledge — it couldn't and doesn't try to eliminate the linkability explicitly introduced by users. For example, a user may post photos of himself or exhibit very identifiable purchasing behavior in two different apps, thus establishing linkability. This type of linkability is app-specific, domain-dependent and beyond the control of `LinkDroid`. Identifiability or linkability of these domain-specific usage behaviors are of particular interest to other areas, such as anonymous payment [25], anonymous query processing [18] and data anonymization techniques.

The list of identifying information we considered in this paper is well-formatted and widely-used. These ubiquitous identifiers contribute the most to information aggregation, since they are persistent and consistent across different apps. We didn't consider some uncommon identifiers, such as walking patterns and microphone signatures, because we haven't yet observed any real-world adoption of these techniques by commer-

cial apps. However, LinkDroid can easily include other types of identifying information, as long as a clear definition is given.

DLG introduces another dimension — linkability — to privacy protection on mobile OS and has some other potential usages. For example, when the user wants to perform a certain task in Android and has multiple optional apps, the OS can recommend him to choose the app which is the least linkable with others. We also noticed some interesting side-effect of LinkDroid's unlinkable mode. Since unlinkable mode allows users to enjoy finer-grained (session-level) unlinkability, it can be used to stop a certain app from continuously identifying a user. This can be exploited to infringe the benefits of app developers in the case of copyright protection, etc. For example, NYTimes only allows an unregistered user to read up to 10 articles every month. However, by restarting the app in unlinkable mode in each session, a user can stop NYTimes from linking himself across different sessions and bypass this quota restriction.

8 Conclusion

In this paper, we addressed the privacy threat of unregulated aggregation from a new perspective by monitoring, characterizing and reducing the underlying linkability across apps. This allows us to measure the potential threat of unregulated aggregation during runtime and promptly warn users of the associated risks. We observed how real-world apps abuse OS-level information and IPCs to establish linkability, and proposed a practical countermeasure, LinkDroid. It provides runtime monitoring and mediation of linkability across apps, introducing a new dimension to privacy protection on mobile device. Our evaluation on real users has shown that LinkDroid is effective in reducing the linkability across apps and only incurs marginal overheads.

Acknowledgements

The work reported in this paper was supported in part by the NSF under grants 0905143 and 1114837, and the ARO under W811NF-12-1-0530.

References

[1] 2013: a look back at the year in acquisitions. http://vator.tv/news/2013-12-07-2013-a-look-back-at-the-year-in-acquisitions.

[2] AGGARWAL, G., BURSZTEIN, E., JACKSON, C., AND BONEH, D. An analysis of private browsing modes in modern browsers. In *Proceedings of the 19th USENIX conference on Security* (2010), USENIX Association, pp. 6–6.

[3] Angry birds and 'leaky' phone apps targeted by nsa and gchq for user data. http://www.theguardian.com/world/2014/jan/27/nsa-gchq-smartphone-app-angry-birds-personal-data.

[4] BACKES, M., KATE, A., MAFFEI, M., AND PECINA, K. Obliviad: Provably secure and practical online behavioral advertising. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2012), SP '12, IEEE Computer Society, pp. 257–271.

[5] BAMIS, A., AND SAVVIDES, A. Lightweight Extraction of Frequent Spatio-Temporal Activities from GPS Traces. In *IEEE Real-Time Systems Symposium* (2010), pp. 281–291.

[6] BUGIEL, S., HEUSER, S., AND SADEGHI, A.-R. Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In *Presented as part of the 22nd USENIX Security Symposium* (Berkeley, CA, 2013), USENIX, pp. 131–146.

[7] DAVIDSON, D., AND LIVSHITS, B. Morepriv: Mobile os support for application personalization and privacy. Tech. rep., MSR-TR, 2012.

[8] EGELE, M., KRUEGEL, C., KIRDA, E., AND VIGNA, G. Pios: Detecting privacy leaks in ios applications. In *NDSS* (2011).

[9] ENCK, W., GILBERT, P., CHUN, B.-G., COX, L. P., JUNG, J., MCDANIEL, P., AND SHETH, A. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI* (2010), vol. 10, pp. 255–270.

[10] FAWAZ, K., AND SHIN, K. G. Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 239–250.

[11] GOLLE, P., AND PARTRIDGE, K. On the anonymity of home/work location pairs. In *Proceedings of Pervasive '09* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 390–397.

[12] HAN, S., JUNG, J., AND WETHERALL, D. A study of third-party tracking by mobile apps in the wild. Tech. rep., UW-CSE, 2011.

[13] HORNYACK, P., HAN, S., JUNG, J., SCHECHTER, S., AND WETHERALL, D. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In *Proceedings of the 18th ACM conference on Computer and Communications Security* (2011), ACM, pp. 639–652.

[14] JEON, J., MICINSKI, K. K., VAUGHAN, J. A., FOGEL, A., REDDY, N., FOSTER, J. S., AND MILLSTEIN, T. Dr. android and mr. hide: fine-grained permissions in android applications. In *Proceedings of Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices* (2012), ACM, pp. 3–14.

[15] KRUMM, J. Inference attacks on location tracks. In *Proceedings of the 5th international conference on Pervasive computing* (Berlin, Heidelberg, 2007), PERVASIVE'07, Springer-Verlag, pp. 127–143.

[16] LABITZKE, S., TARANU, I., AND HARTENSTEIN, H. What your friends tell others about you: Low cost linkability of social network profiles. In *Proc. 5th International ACM Workshop on Social Network Mining and Analysis, San Diego, CA, USA* (2011).

[17] LEE, S., WONG, E. L., GOEL, D., DAHLIN, M., AND SHMATIKOV, V. πbox: a platform for privacy-preserving apps. In *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation* (2013), USENIX Association, pp. 501–514.

[18] MOKBEL, M. F., CHOW, C.-Y., AND AREF, W. G. The new casper: query processing for location services without compromising privacy. In *Proceedings of the 32nd international conference on Very large data bases* (2006), VLDB Endowment, pp. 763–774.

[19] NARAYANAN, A., AND SHMATIKOV, V. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on* (2008), IEEE, pp. 111–125.

[20] NAUMAN, M., KHAN, S., AND ZHANG, X. Apex: extending android permission model and enforcement with user-defined runtime constraints. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security* (2010), ACM, pp. 328–332.

[21] PEARCE, P., FELT, A. P., NUNEZ, G., AND WAGNER, D. Addroid: Privilege separation for applications and advertisers in android. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security* (2012), ACM, pp. 71–72.

[22] SHEKHAR, S., DIETZ, M., AND WALLACH, D. S. Adsplit: separating smartphone advertising from applications. In *Proceedings of the 21st USENIX conference on Security symposium* (2012), USENIX Association, pp. 28–28.

[23] STEVENS, R., GIBLER, C., CRUSSELL, J., ERICKSON, J., AND CHEN, H. Investigating user privacy in android ad libraries. *IEEE Mobile Security Technologies (MoST)* (2012).

[24] TRIPP, O., AND RUBIN, J. A bayesian approach to privacy enforcement in smartphones. In *Proceedings of the 23rd USENIX Conference on Security Symposium* (Berkeley, CA, USA, 2014), SEC'14, USENIX Association, pp. 175–190.

[25] WEI, K., SMITH, A. J., CHEN, Y.-F., AND VO, B. Whopay: A scalable and anonymous payment system for peer-to-peer environments. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on* (2006), IEEE, pp. 13–13.

[26] XIA, N., SONG, H. H., LIAO, Y., ILIOFOTOU, M., NUCCI, A., ZHANG, Z.-L., AND KUZMANOVIC, A. Mosaic: quantifying privacy leakage in mobile networks. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM* (2013), ACM, pp. 279–290.

[27] Xprivacy - the ultimate, yet easy to use, privacy manager for android. `https://github.com/M66B/XPrivacy#xprivacy`.

[28] XU, R., SAÏDI, H., AND ANDERSON, R. Aurasium: Practical policy enforcement for android applications. In *Proceedings of the 21st USENIX conference on Security symposium* (2012), USENIX Association, pp. 27–27.

[29] ZANG, H., AND BOLOT, J. Anonymization of location data does not work: a large-scale measurement study. In *Proceedings of MobiCom '11* (New York, NY, USA, 2011), ACM, pp. 145–156.