Multi-Device Context-Sensitive Attacks Against Privacy

Edgardo A. Barsallo Yi
Delivery Dynamics
Panama, Panama, Panama
edgardo.barsallo@deliverydynamics.com

Joshua D. O. Majors
Purdue University
West Lafayette, IN, USA
jmajors@purdue.edu

Aditya Vardhan Padala
Purdue University
West Lafayette, IN, USA
padalaa@purdue.edu

Darren Wu
Purdue University
West Lafayette, IN, USA
wu1797@purdue.edu

Aravind Machiry
Purdue University
West Lafayette, IN, USA
amachiry@purdue.edu

Saurabh Bagchi
Purdue University
West Lafayette, IN, USA
sbagchi@purdue.edu

Abstract

As the adoption of wearable and smart devices increases, their privacy and security are still a concern. These devices collect sensitive data and constantly communicate with each other, posing new privacy threats that need to be understood and addressed. In this paper, we analyze the privacy of smart devices from a *multi-device* perspective. The central premise of our work is that information available at each device may be non-sensitive or lightly so, but by orchestrating information from multiple connected smart devices, it is possible to infer sensitive content. To verify this, we conduct a user study to understand user perceptions towards privacy on smart devices and contrast them with their actual behavior while operating these devices. We then present an attack framework that can leverage tightly coupled and connected smart devices, such as mobile, wearable, and smart TV, to leak sensitive information inferred from individually non-sensitive data. Finally, we introduce a tool based on NLP techniques to identify potential privacy vulnerabilities on smart devices and propose an integrated solution to increase smart devices' security. This analysis helps close the gap between user's perception and reality regarding privacy risks within their smart ecosystem.

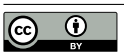
CCS Concepts

• **Security and privacy** → **Mobile platform security**; **Privacy protections**.

Keywords

smart devices, wearables, privacy, security

ACM Reference Format:
Edgardo A. Barsallo Yi, Joshua D. O. Majors, Aditya Vardhan Padala, Darren Wu, Aravind Machiry, and Saurabh Bagchi. 2025. Multi-Device Context-Sensitive Attacks Against Privacy. In *Proceedings of the Fifteenth ACM Conference on Data and Application Security and Privacy (CODASPY '25), June 4–6, 2025, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3714393.3726508



CODASPY '25, Pittsburgh, PA, USA

ACM ISBN 979-8-4007-1476-4/2025/06
https://doi.org/10.1145/3714393.3726508

1 Introduction

There has been a significant amount of privacy research on smartphones and, to a more limited extent, wearables and smart devices. The research has identified vulnerabilities (in phones [10, 41] and wearables [29, 30]), and some of it has presented mitigations (in phones [9, 12, 39] and wearables [11, 16]). Broadly speaking, the vast majority of the prior work has taken a single platform or a single device view of user privacy. By this, we mean that the analysis has been into privacy implications of information coming *from a single device* — this could be single or multiple sources of information (i.e., multiple apps within the same device), either covert or overt (i.e., user-visible) sources. In a small fraction of the cases, the work has aggregated information from multiple devices, though in very different contexts, such as devices of the same kind, and therefore naturally belonging to multiple users [71] or for data transformations among multiple IoT devices [33]. We posit that the convergence of trends laid out above about multiple smart devices of a single user and their interactions should lead us to look at privacy in a new light.

An important aspect of privacy is context, as is widely acknowledged [12, 58]. For example, a smartphone camera taking high-def video is acceptable when the user is in a public space, while that camera should lower resolution if say the user is on a call from her home, and further, in more private spaces, the camera should never be activated. The use of context to drive privacy decisions and then to create tools to enforce such decisions has been a fruitful line of work [12, 48, 66]. For example, the early work in ipShield [12] allowed a smartphone user to set context-aware fine-grained privacy rules. It inferred privacy-relevant context by monitoring of the phone sensors accessed by an app and used this information to perform a privacy risk assessment, which was then used to decide whether to allow or disallow an action. A key missing piece to this use of context has been that the context can come from multiple devices owned by a user. This is a natural outgrowth of the trend outlined above, multiple devices owned by a user and autonomous interactions among them.

The nature of these devices is different, and the kinds of context they provide are also distinct and often complementary. For example, physiological signals are available from the sensors on a smartwatch or a fitness tracker, streaming media viewing information is available from a smart TV, and location information is available from a smartphone. Although individually non-sensitive or only lightly sensitive, these information components can lead

to privacy violations when combined across multiple devices in certain contexts. Consider a music streaming application on a smartphone that uses the user's location to access local radio stations. The application may also use user's whereabouts for location-based advertising and push ads from local stores. This is already a clear privacy violation. Now let us consider the privacy implication in a multi-device context. If the streaming music app detects that the user is in a leisure or recreational place (e.g., bar or restaurant), it can collude with other devices' apps to target the user's privacy. The user's smartwatch collects her heart rate, and another app on the mobile that uses the microphone now performs automatic gender recognition based on the voices. Using these two sources of information, it is possible to infer the user's emotions [28, 55] towards friends and acquaintances and infer social or dating preferences of the user. This work has two questions it seeks to answer.

(1) Can information collected from multiple devices of a user be used to infer sensitive information, and therefore lead to amplified privacy attacks, compared to single-device scenarios?
(2) Can context help in deciding when to orchestrate such individual pieces of information?

The question of using context to launch privacy attacks (based on multi-device information) is significant from the point of view of the stealthiness of the attack. Collecting information on a device consumes resources, such as performing GPS readings, which is well known to be energy hungry. Then, aggregating such information across multiple devices as well as performing inferencing on the combined information can again be resource intensive. Therefore, triggering the attack based on context can reduce the resource consumption of the attack and make it more stealthy. The road to answer the above questions leads to the following contributions.

- We conduct a university IRB-approved user survey to elicit inputs about their main privacy concerns on multiple connected personal devices. Our survey brings out several hitherto unknown privacy expectations and concerns. An automated tool developed by us shows how far the user's privacy perception matches with the permissions she has granted on her devices.
- We create a multi-device attacks framework, ***MadCap*** (**M**ultiple **A**ggregated **D**evice **C**ontext **A**ware **P**rivacy). We implement three previously proposed privacy attacks using this framework, each of which uniquely leverages the interactions of smart devices. The proposed attack is an "amplification" attack in that it collects data from different platforms that are individually not sensitive or lightly so, and uses them to infer sensitive information.
- We introduce ***PerQry***, a tool that uses NLP techniques to answer user queries about her privacy sensitivities and then analyzes permissions granted to apps on her device. This aids as a privacy check tool telling the user what privacy attacks are possible or not possible, including in multi-device contexts. Finally, we show how this tool can be integrated to increase the security and privacy in multi-device contexts.

The actionable insights that we gather from our adversarial analysis in this paper are two-fold. First, the OSes on the smart devices should allow for coordination not just for functionality (as is done today), but also to enforce fine-grained privacy controls that span multiple devices. Second, from a usability standpoint, we need tools that can model the context that arises from multiple devices. This will enable users to make context-based privacy decisions or automate the process through rules that they have pre-installed.

2 Background

Android Permissions. Permissions form one of the fundamental aspects of the Android ecosystem in restricting an app's access to data (e.g.,contact list) and actions (e.g., sending an SMS) [21]. Android considers users' data and certain system resources as sensitive and allows users to control an app's access to them. An app that requires access to these restricted data or actions should request by specifying corresponding permissions its `AndroidManifest.xml` file and, in a few cases, should also get authorization from the user at runtime. Android classifies permissions based on the sensitivity of the resources they protect as follows:

(1) **Normal** permissions control access to those resources that do not pose a major risk to the users' privacy or the device's operation. These permissions will be granted automatically *without* confirmation from the user. e.g., `INTERNET`.
(2) **Signature** permissions are similar to normal permissions, but with the restriction that an app requesting permission must be signed with the same signature as the app that defines the corresponding permission. e.g., `READ_VOICEMAIL`.
(3) **Dangerous** permissions control resources that could potentially affect the users' privacy or the device's operation. The user must grant these permissions explicitly at runtime. e.g., `SEND_SMS`
(4) **Special** permissions depend on the device manufacturers or OEMs. e.g., `WRITE_SETTINGS` on Google phones.

Privacy Risk on Android Ecosystem. Android enforces its permissions and other restrictions at app level, treating each app independently without considering risks from collaboration with other apps [1]. Previous studies show that user is more cautious [22, 23] in installing an app with multiple permissions (such as `INTERNET` and `LOCATION`) than installing multiple apps with same cumulative permissions (e.g., two apps, one with with `INTERNET` and the other with `LOCATION`). This further motivates attackers to focus on creating multiple colluding apps than a single malicious app. In Android, precisely analyzing inter-app communications is a known hard problem because of the binder IPC interface [63]. Other communication channels [31], such as storage, further exacerbate the problem. This problem becomes intractable when the apps are across different devices, such as smartphone and wearables. First, the devices could be running different versions of the Android OS. Second, as these devices differ in their capabilities, the OS design differs between these devices [8, 37]. Having a synchronized privacy policy that spans across multiple devices running different OSes is an open problem [54]. Our work further explores the privacy concerns that could arise when apps across multiple devices collude.

3 Overview

The significant amount of productive research on the privacy of mobile platforms has treated each user device in an isolated context. The fact that we increasingly own multiple smart devices and that these devices are communicating with each other on a continual basis introduces new privacy scenarios. The fundamental premise

Table 1: Sample set of permission and their respective likelihood of app installation and security perception.

Permission	Type	(a) Likelihood of installation across categories						(b) Security perception	
		Music & Audio	Education	Business	Productivity	Health & Fitness	Overall	Risky	Safe
Full network access	Normal	88.06%	79.10%	89.55%	85.07%	76.12%	83.58%	49.25%	46.27%
Allows the app to configure the local Bluetooth	Normal	67.16%	47.76%	59.70%	55.22%	76.12%	61.19%	35.82%	58.21%
Access to your location	Runtime	65.67%	56.72%	62.69%	62.69%	74.63%	64.48%	61.19%	37.31%
Access to your device storage	Runtime	80.60%	65.67%	67.16%	73.13%	55.22%	68.36%	41.79%	52.24%
Access sensor data about your vital signs	Runtime	41.79%	40.30%	46.27%	49.25%	74.63%	50.45%	49.25%	44.78%
Take pictures and record videos	Runtime	43.28%	43.28%	76.12%	58.21%	41.79%	52.54%	62.69%	37.31%
Record audio	Runtime	52.24%	49.25%	68.66%	55.22%	40.30%	53.13%	59.70%	32.84%
Read phone status and identity	Signature	44.78%	43.28%	55.22%	47.76%	53.73%	48.96%	61.19%	34.33%
Enable accessibility service and give full control	Signature	49.25%	35.82%	40.30%	38.81%	41.79%	41.19%	65.67%	26.87%

behind our work is that such automated, i.e., non user-mediated connections among the smart devices pose new vulnerabilities by allowing an adversary to piece together multiple non-sensitive (or lightly sensitive) attributes into sensitive attributes.

3.1 Problem Context

Nowadays, the ubiquitous connectivity and context-awareness nature of wearables pose new challenges. Smart devices constantly communicate with each other. Not only do wearables depend on a mobile device [8, 56], but mobiles are also frequently used to control IoT devices or OTT (Over The Top) platforms, such as Google TV and Amazon Fire TV [19, 43]. Prior works have demonstrated how a mobile app can use sensing data collected from a device to infer sensitive information from the user [12, 35, 68]. However, the implicit connection between smart devices enables the possibility of multiple attackers colluding to leak data from devices with the ultimate objective of inferring sensitive information from the user. Furthermore, the different nature of smart devices broadens the range of possible inferences by allowing attacks that would not be possible by targeting a single device. Our central premise is that by collecting information specific to a particular type of device, such as health data available on wearable devices, and then strategically integrating such multimodal data or user interactions with her other devices, it is possible to craft a richer set of attacks. Such attacks are either infeasible or less damaging in individual device settings.

3.2 User Perception and Reality

As a motivation for our work, we decided to contrast user perception and reality regarding privacy risk in multi-platform scenarios. To this end, we designed a user study that was approved by the Institutional Review Board of our home institution. First, we created an online questionnaire and distributed it to 93 participants, drawn from Amazon Mechanical Turk (AMT) and university students. The objectives of our survey were to determine:

(1) users' knowledge regarding the Android permission model;
(2) users' attitudes and behaviors towards security while installing applications on their devices;
(3) users' perception of privacy risks in scenarios that involve multiple devices.

Then, we collected a list of the apps installed and the respective permissions granted and denied from participants' compatible devices. This information helps us to get a reality check on what permissions the participants have granted and compare it against their answers in the online survey. Regarding the above stage, users were aware of the role of the scanning app they were installing. Our survey revealed that participants care about privacy risks in these new scenarios. Moreover, the results evidence that users' behavior for the same permissions varies across devices. We discuss in further detail our findings in the next section.

4 User Study

As discussed in section 2, users may tend to overlook the privacy risks associated with using tightly coupled devices, such as mobiles and wearables. We conducted a user study to understand the users' perceptions and behavior related to privacy.

4.1 Methodology

The study consisted of an online survey and a one-time collection of data from participants' mobile and wearable devices.

4.1.1 Online Survey. The online survey consisted of four sections. The first two sections included questions to classify the participants according to demographics and their knowledge and expertise in using smart devices. The last two sections focused on participants' behavior while installing apps and their perceptions of the possibility of privacy infringement expectations in various scenarios. All the sections included control questions to validate the quality of the responses. From the subsequent analysis, we removed the submissions that failed to answer the control questions correctly.

4.1.2 Permissions Granted on User Devices. Additionally, for collecting the apps installed and the list of permissions granted or denied, we developed an Android scanning app (Permeso) and its respective companion app in the wearable. We asked the participants to install Permeso app on their compatible devices and upload the data collected by the app. Since we only need to collect the data once, we requested the participants to uninstall the app after finishing the study.

4.2 Results

In all, we had 93 survey participants, and Permeso collected information on 4,365 apps. The analysis presented here is for the 5 most popular categories, which covered 369 apps (out of the 4,365). The survey participants came from AMT, and students at the two universities were represented by the co-authors of this submission.

4.2.1 Permission and Categories. One of the primary goals of our user study was to understand users' attitudes toward the Android security model and examine the impact of permissions requested by an app on users' decision to install it on their device. To that end, we chose five of the top categories from the Google Play Store based on the number of apps [5]. Then, we selected a sample set of 9 of the most frequently requested permissions across categories in Android and Wear OS, as shown in Table 1. Here, we do not select any special permission in the sample set since these permissions are only requested by the system and OEM's applications. In the survey, we asked participants how likely they would install an application, from each of the five top categories, if it requires a permission defined in the sample set. Table 2 summarizes the survey responses for each category and permission type, while the same information is broken down by individual permission in Table 1(b).

Table 2: Likelihood of installation of the application across Permission Type.

Category	Normal	Runtime	Signature
Music & Audio	77.61%	56.72%	47.01%
Education	63.43%	51.04%	39.55%
Business	74.63%	64.18%	47.76%
Productivity	70.15%	59.70%	43.28%
Health & Fitness	72.39%	57.31%	47.76%

As shown in Table 2, the category of an application has an impact on user decision whether to grant or deny permissions. As expected, participants are more likely to grant normal permissions, with rates reaching as high as 77.61% (Music & Audio) [1]. Moreover, as can be seen from Table 1, the majority of the respondents (83.58%), did not seem to have a reservation in granting the Full network access permission. Again, the rate is higher in Music & Audio (88.06%) and Business (89.55%) categories. This may be explained by the fact that trusted music players and web conference tools (e.g., Zoom, Skype) are prevalent applications in these categories. Survey respondents seem more cautious with "Education" category apps, perhaps because they have less experience with apps from widely popular and trusted brands in that category. Participants acknowledged being less willing to install apps that request signature permission. However, the rate of 47% can still be considered high since malwares or attackers often abuse signature permissions.

4.2.2 Security Perception. We also evaluated how safe the user considers each of the permissions defined in our study. Table 1(b) summarizes the results. As expected, participants consider *signature* permissions riskier than the rest. Also, most of the participants considered *normal* permissions as safe, which stands to reason as these permissions are granted without users being prompted. However, an interesting finding is that:

[1]Note that the respondents were not informed by us on the definitions of "Normal", "Runtime", and "Signature" permissions, and neither were they told which category each permission belonged to. Thus, their responses can be taken to reflect no bias from such knowledge. The time taken to complete each survey question also makes it unlikely that the respondents were searching for the category of each permission prior to answering.

Finding 1: Dangerous permissions, such as *Access to your device storage*, were considered safe by the majority of the participants.

Recall that the respondents are making their decisions without being aware of which category each permission belongs to. We categorize it for the purpose of the analysis. The following section delves into these findings to see how much gap there exists between the survey responses and the actual permissions the users have granted on their devices.

Table 3: Percentage of participants whose responses differed across mobile and wearable app. For clarity, we include the four permissions with the most significant differences.

Category	Body Sensors	Storage	Network	Location
Music & Audio	8.82 %	5.88 %	17.65%	32.35%
Education	14.71 %	11.76 %	23.53%	23.53%
Business	8.82 %	17.65 %	17.65%	17.65%
Productivity	14.71 %	11.76 %	17.65%	26.47%
Health & Fitness	8.82 %	8.82 %	20.59%	23.53%
Overall	26.47 %	29.41 %	35.59%	47.06%

4.2.3 Permission Behavior in Multi-Device Scenarios. Finally, we assessed participants' permission behavior on multiple platforms. Thus, we asked the same questions, but this time with respect to wearables apps instead of mobile. Table 3 shows the contrast between participants' responses on mobile versus wearable apps, specifically for the top four permissions based on the discrepancy: *access sensor data about vital signs*, *access to device storage*, *full network access*, and *access to location*.

Here, 44.1% of participants expressed being more conservative when installing apps on wearable devices. These individuals responded that they would not install an app on a wearable even though it requests the same permission they will grant to a mobile app. Hence, this means that for users, the risk associated with permission varies depending on the platform.

Finding 2: Our results indicate that 44.1 % of participants declared to be more conservative while granting permissions on the wearable. Further, the most frequent differences were on "Full network access" and "Access to location" permissions.

4.3 Perception vs. Reality

We also assessed how the answers given by the user match with their actual behavior while installing apps on their devices. We compared the data collected by Permeso with participant's responses for each of the categories included in the study. We were particularly interested to see if there are discrepancies between survey responses and the user's actual behavior. For example, *a respondent may claim that she would not install music applications that require "Read phone status and identity" permission. However, she has granted that same permission to apps in that category on her phone.*

Table 4: Discrepancy between responses and permission collected by Permeso.

Category	Perception: Will Grant Reality: Not Granted	Perception: Will Not Grant Reality: Granted
Music & Audio	28.0%	59.7%
Education	37.4%	35.5%
Business	34.4%	47.2%
Productivity	36.2%	64.5%
Health & Fitness	47.3%	48.7%

Table 4 shows the breakdown for each category, where a higher percentage indicates a greater discrepancy. The number in the second column indicates cases where users are more conservative (i.e., strict) in reality compared to their (more lax) perception. This is surprising but not risky as users are more strict in reality.

The numbers in the third column are interesting, where users are more lax in reality compared to their perception. In other words, these numbers show the cases where users are lenient in granting permissions compared to their perception of being strict. The higher numbers in the third column compared to the second indicate that users are laxer in reality than in their perception. The values are high for the Music & Audio and Productivity categories, indicating that our desire for entertainment in personal life and productivity gains in professional life could make us more privacy lenient. For business apps, which traditionally tend to request a high number of permissions, these results are disturbing since an attacker can disguise the malicious application as a business application. Based on the data collected through Permeso, business apps requested on average more permissions (28) than the rest of the categories analyzed, except for productivity apps (31). In a broader examination across all Android application categories, business apps rank within the top 10, following behind categories such as communications (44), tools (42), photography (39), personalization (31), and shopping (29).

Finding 3.a: In contrast to users perception, in reality users are more lenient in granting permissions. The discrepancy is non-uniform, and there are certain categories of apps where users are more lenient compared to others.

4.3.1 Discrepancy v/s Apps Popularity. Our survey only uses the category of apps without mentioning a specific app or brand name. However, Permeso examines the apps the user has installed; many of these apps are presumably from vendors that are trusted. This might pose a familiarity bias threat to the validity of our findings. Specifically, users might be willing to grant permissions to widely popular, and therefore implicitly more trusted, apps from well-known brand names.

To understand this, we delved into the details of the popularity of applications that Permeso detects on the user devices; popularity is measured by the number of downloads from the Google Play Store, as is typically done. Figure 1 shows the difference between the survey and data collected using Permeso grouped by the number of installs of the application. From the heat map, we see that the user behavior for granting permission (i.e., second column) does not seem to be heavily impacted by the popularity of the apps. In most cases, the values hover around the 50% mark.

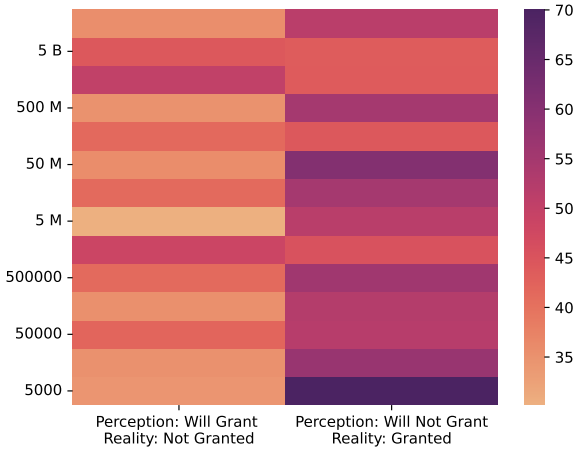


Figure 1: Relation between popularity of the app and discrepancy between the survey responses and actual permissions granted by the user on her devices.

Finding 3.b: Our results show that apps' popularity does not influence the discrepancy between the user's perception and reality.

4.4 Privacy Related Scenarios

We also asked users for their opinions on the privacy risk in various scenarios involving multiple devices. Previous research [24] has only focused on scenarios in single device context, i.e., either mobiles or wearable devices, but not both. Specifically, we crafted 16 scenarios where it was possible to infer sensitive information by using data from multiple devices, especially mobile and wearable. Each scenario succinctly describes a multi-device privacy risk, i.e., using data from multiple devices to infer sensitive information, which will be leaked to directly related entities or unrelated entities (e.g., advertisement companies). We asked our participants to classify each scenario as "risky" or "safe" regarding privacy. We also included an additional category, "Do not know", to indicate a neutral position. The Table 5 lists all the scenarios and the percentage of users who considered each scenario as risky.

Finding 4: Participants seem more concerned about scenarios where their data is explicitly shared with entities directly related to them, such as HR and insurance companies. In contrast, scenarios where the data is either leaked publicly or used for advertisement do not concern them as much.

Users perceived scenarios to be relatively safe when scenarios involved an app suggesting something to the user *without* explicit mention of an external party becoming aware of that information. Obviously, this situation hides the fact that the *app* knowing and suggesting something could very well mean that the *creator of the app* and any third parties that the creator has signed deals with can access the user data.

Table 5: Percentage of participants who responded 'yes' to whether each scenario was considered a violation of their privacy

Permission	Risky
A productivity app on the mobile collects data about the music listened to by a user and, using her wearable locations, determines she is at work. The app reports the music genres to the HR department.	68.66%
Using the wearable's location, an app installed on the mobile device can suggest the apps a user interacts with at work. The app reports the list to the user's manager.	65.67%
Using sleep data collected from a smartwatch and the mobile's location, an app can suggest medical disorders in the users. The app could disclose this information to the user's health insurance company.	65.67%
Based on wearable fitness data and other sensors on the mobile, an app determines the user's grocery patterns. The app could disclose this information to third parties.	59.70%
Based on fitness tracker data, an app can infer if a user has any heart pre-condition. The app could disclose this information to the user's health insurance company.	58.21%
Based on variations on your heart rate while readings news from the mobile, an app can suggest your political preferences.	55.22%
Using the microphone in your mobile and variations in your heart rate, an app can suggest your personal preferences in dating.	52.24%
Using the mobile device's location and the sensors installed on a smartwatch, an app can suggest the location of ATMs commonly visited by the user.	52.24%
Using the mobile device's location and the sensors installed on a smartwatch, an app can suggest the user's preferred offline work locations.	41.79%
A shopping app recommends a user to buy merchandise of the musical artists she often listens to while working out, using the fitness data from the wearable and the microphone on the mobile.	38.81%
Data collected from the user's device (mobile's location and wearable's fitness data) suggest the restaurants she usually frequents to eat.	37.31%
An app installed on a user's phone could tell her daily commutes based on the fitness data collected from a fitness tracker or smartwatch.	35.82%
Based on data collected from sensors on the mobile and the wearable, an advertiser shows local stores' ads within a user's daily commutes.	32.84%
Using the mobile's location, a shopping app can show sale offers on energy drinks from nearby stores if it detects a user is working out based on the wearable's fitness data.	28.36%
Data collected from the user's phone and her smartwatch by an app can suggest the user's preferred times for working out.	28.36%
An educational app, either on a mobile or wearable device, could tell the user location based on the fitness tracker data and customize the app's experience depending if it is an outdoor or indoor area.	26.87%

4.5 Summary

Our findings shed light on certain important aspects and suggest directions for effective privacy protection measures. Specifically, the well-known problem [42, 50] of users lacking knowledge of the severity of certain permissions still exists (**Findings 1, 2**). Users' perception and their practices regarding permission vary (**Finding 3.a**). We urge the community to pay special attention to survey-based user studies involving permissions, as the results might not truly reflect users' real behavior. Privacy-related scenarios (**Finding 4**) might be an effective way to warn about potential misuse of permissions by an app. Finally, *there is the potential for multi-device context-sensitive attacks, whose severity may not be correctly assessed by the users (**Findings 1, 2, 3.a, 3.b, 4**).*

5 Our Attack Framework: MadCap

We designed MadCap, a framework aiding the implementation of *stealthy multi-device context-sensitive attacks*. Table 6 summarizes a (limited) list of attacks that can be crafted following our framework. We implemented three Proof of Concept (PoC) attacks using MadCap, demonstrating the generality of our framework.

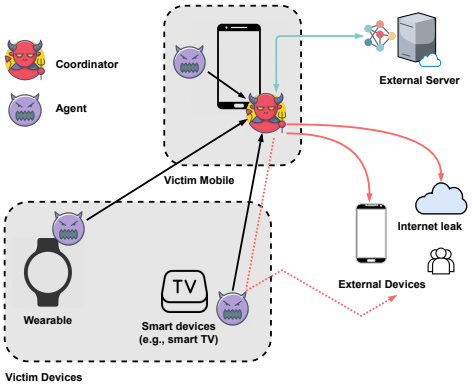


Figure 2: Architecture of the MadCap framework for multi-device privacy attacks.

5.1 Adversary Model

Attacker. An attacker is a developer who publishes a malicious app in the Google Play Store. These apps behave as benign at the start of their life cycle. However, after achieving user trust that it serves a legitimate purpose, it can start the attack stealthily and collude with other apps in the user's devices to leak sensitive information.
Attacker Goal. We assume that the adversary intends to leak sensitive information from the user by collecting individually non-sensitive (or lightly sensitive) data from multiple platforms, such as mobile, wearable, and smart TV. The attacker's malicious apps collect data from the victim's devices and use it to infer and leak sensitive information about the user. We refer to sensitive information as any knowledge of the user that must be protected against unauthorized access since its disclosure could cause adverse consequences, such as damage to her financial standing, reputation, employability, or insurability.
Victim. A victim is a smartphone user who has her phone paired with a wearable device or uses the phone to interact with other smart devices. The victim installs malicious apps on their devices from regular channels, such as Google Play Store.
Feasibility. We found that 64% of the top 200 popular wearable apps in the Google Play Store have a counterpart in the mobile device. Therefore, it is quite conceivable that if the user is socially engineered to download an app on her wearable, she is likely to download the (seemingly benign) companion app on her mobile.

5.2 Attack Overview

We first present an overview of our attack named MadCap. As shown in Figure 2, MadCap exploits the previously described attack approach by orchestrating a collusion attack between the apps installed on multiple platforms.

5.2.1 Architecture. MadCap consists of two components: *agents* and *coordinator*. Both components are malicious apps, but their objectives vary in the overall attack scheme. These components

Table 6: Summary of privacy attacks that can be implemented using our MadCap framework.

	Attack Aim	Mobile	Wearable	Android TV
A	Tracking user's workout	Location	Heart rate,Accelerometer, Gyroscope	-
B	Tracking user's music preferences	Location, Microphone	Heart rate,Accelerometer, Gyroscope	-
C	Tracking user's activity	Accelerometer, Gyroscope	Heart rate,Accelerometer, Gyroscope	-
D	Leak conversation during meetings	Calendar	Microphone	
E	Tracking virtual work place	Location	Accelerometer, Gyroscope	-
F	Tracking sleeping patterns	Location	Accelerometer, Gyroscope	-
G	Show unsolicited ads	Location	Heart rate,Accelerometer, Gyroscope	-
H	Disclose contextual information	Location, Accelerometer, Gyroscope, Camera, Mic	Heart rate, Accelerometer, Gyroscope	-
I	Disclose ATM's PIN sequence	Location	Acceloremeter, Gyroscope, Magnometer	-
J	Personal preference in dating	Location, Device Storage, Microphone	Heart rate	-
K	Political preference (based on content watched)	-	Heart rate	Accessibility Service

* We are assuming the data is leaked using conventional channel such as Internet, which does require a normal permission.

leverage the permissions legitimately given by the user to collect individually non-sensitive or only lightly sensitive data. Once the data is collected, the agents forward it to the coordinator, usually installed on the mobile device due to energy constraint limitations of smart devices, such as wearables. Then, the coordinator infers some sensitive information using the data collected. In cases where the inference algorithms require more computation power, this mission can be accomplished by an external server. Finally, the coordinator can leak the information itself, or it can delegate the task to an agent. The communication between components can happen via Wear OS Data Layer API, WiFi, or Bluetooth.

5.2.2 Trigger Conditions. Neither the agents nor the coordinator is expected to collect data from the user continuously; instead, the attack needs to be triggered if specific conditions are met (i.e., *Context*). By doing this, we keep the attack as stealthy as possible and avoid wasting scarce energy resources on the devices so as not to raise the victim's awareness. The trigger conditions vary depending on the sensitive information that is targeted. For example, if MadCap aims to leak a user's ATM pin number, it will track the user's location sporadically. When it detects that the user location matches with a nearby ATM, coordinator will trigger the attack, and the agents will start collecting the required sensing data needed to infer the victim's ATM PIN.

5.3 PoC Attacks

We demonstrate the generality of MadCap framework by implementing three multi-device attacks. We selected the attacks based on their severity. We emphasize that the general principle remains the same across all attacks, i.e., inferring privacy-sensitive data from non-privacy-sensitive data across multiple devices.

5.3.1 Attack #1: Revealing ATM PIN sequence. The attack focuses on inferring the victim's ATM PIN. The attacker requires a coordinator on the victim's mobile device and an agent deployed on the wearable. The coordinator, which maintains a list of ATM places near the victim, uses the `LOCATION` permission to keep track of her whereabouts. As soon as the coordinator detects the victim is close to any ATM location registered, it triggers the attack by communicating with the agent. Then, the agent uses the wearable's sensing data to infer the victim's hand movements and suggest a possible ATM PIN. Prior works [60, 61] already shown the feasibility of inferring a user's personal PIN sequence by using sensing data from the accelerometer, gyroscope, and magnetometer with high accuracy (80%-90%) without any training and contextual information. It is noteworthy that the agent does not need any permission to access the required sensing data.

For our implementation, we kept the nearby ATM locations in geofences of 10m radius. When the victim enters any of the geofences declared in the coordinator, it will communicate with the agent deployed in the wearable using the Data Layer API. Then, the agent collects the sensing data and sends it back to the coordinator in batches to minimize the communication overhead. Although the Geofences API limits to 100 locations per app, this can easily be overcome by dynamically updating the list.

5.3.2 Attack #2: Inferring Personal Preference in Dating. The goal of the attack is to suggest the gender dating preference of the victim based on the emotions expressed. The attacker requires access to the victim's location (`LOCATION`) and microphone (`RECORD_AUDIO`) in the mobile, and the sensing data of the vital signs (`BODY_SENSORS`) on the wearable. The coordinator tracks the victim's whereabouts using the GPS sensor data. The coordinator triggers the attack when it detects that the victim is located in a place of interest from Google Maps (e.g., restaurants, bars). Then, the agents start collecting data from the microphone and the heart rate sensor. The audio data is used to identify people and their respective gender. Harb *et al.* [27] have shown the feasibility of gender identification with high accuracy (94.2% for males and 88.0% for females) by using audio classifier models speech. Meanwhile, the heart rate sensor data is used to determine the victim's mood as has been demonstrated multiple times [34, 36]. By combining both sources of information, MadCap can determine if the victim feels any affection toward their partner and identify the gender of the person.

In the implementation, we also used geofences to keep track of the places of interest. The communication between the coordinator (mobile) and the agents happens either via Intents (mobile) or through Data Layer (wearable). The coordinator triggers the attack once the victim enters any geofences. Both agents send the collected data to the coordinator in batches to minimize overhead.

5.3.3 Attack #3: Tracking user's workout. This attack focuses on inferring sensitive patterns relative to user workout behavior, such as days and times of the week the victim prefers to work out and preferred locations. The attacker requires access to the heart rate sensor (`BODY_SENSORS`) and accelerometer on the wearable, and the

Table 7: Success ratio per each attack from the dataset of Top 200 apps. Attacks are described in Table 6.

Attack Type	A	B	C	D	E	F	G	H	I	J
% apps vulnerable	13.02	2.60	16.14	1.52	62.5	62.5	13.02	2.08	12.50	5.72
% apps partially vulnerable	65.62	69.79	100.00	16.67	62.5	62.5	65.62	53.12	64.58	73.95

victim's location (LOCATION) on the mobile. The coordinator, installed on the wearable, uses the device's accelerometer to track the user's movements, and upon detecting significant motion, it coordinates with both agents so they can start collecting the sensing data. Prior works [64] have already shown that accelerometer data can be used to identify user physical activity. To further minimize the overhead on the wearable, the agent shutdowns the attack if the heart rate is below 100 bpm. We selected this limit (100 bpm) as it is the lowest bpm in every "moderate exercise" target heart rate range as specified by American Heart Association[7].

Our implementation includes a server to collect the data. The server runs a clustering algorithm to extract the victim's most frequent workout times and determine the workout locations based on averages of the geolocation coordinates. This algorithm requires at least a week of data to infer the workout habits accurately.

6 Attack Evaluation

Although we implemented only three attacks, the principle methodology is the same for all attacks in Table 6. The key difference is the feasibility of acquiring the desired permissions by apps on various devices.

6.1 Feasibility

We evaluate the feasibility of our attacks in terms of the likelihood of acquiring the required permissions. Our dataset included the top 200 wear apps from the Google Play Store and their corresponding companion mobile apps. Out of them, we removed 7 pairs of paid apps, lowering the dataset to 193 apps.

Table 7 shows the feasibility of the 11 different privacy attacks presented in Table 6 across these top apps. Attack K was excluded from our analysis because of the lack of ability to specifically filter Android TV apps. While most of the attacks do not contain every permission required to launch the attack, many apps do ask for partial permissions that are required to execute the attack. Multiple apps have the potential to cooperate in order to perform privacy attacks, not only making the attacks more feasible but also stealthier. It is less likely that the user will grant to a single app many permissions that a user may consider risky. However, the threat arises when the permissions are spread across multiple apps, and relevant to this work, even multiple devices, when the individual apps do not appear to be malicious. The threat is especially exacerbated when multiple devices are used because of the heterogeneity of sensors and correspondingly attack surfaces, on different devices.

Figure 3 depicts the number of possible privacy attacks using the same top 193 apps. The X-axis shows the cumulative number of multi-device privacy attacks (from the 11 in Table 6) that are possible the list. The data point "1" on the X-axis indicates that 100% of the apps have at least 50% of the permissions needed to execute at least one attack, while 65% of the apps grant all the permissions needed to execute at least one attack. While the percentage of apps containing *all* the required permissions drops fairly quickly with increasing the number of attacks, the portion of apps containing *partial* permissions drops much more slowly. This highlights the possibility that attacks can be launched not by an individual app, but by a set of apps that collude and share data. Such collusion among apps can dramatically increase the attack surface.

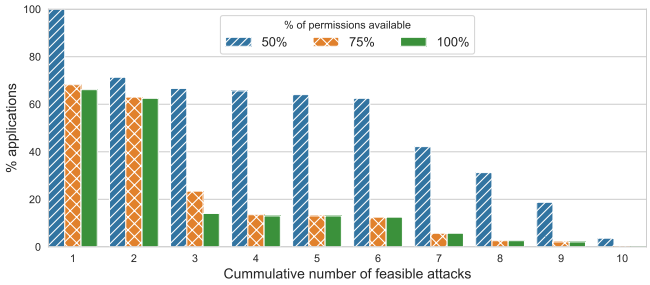


Figure 3: Cumulative number of feasible attacks from dataset of top 200 apps.

6.2 Stealthiness

One of the main indicators of a continuous attack is increased energy consumption, especially on battery-powered devices. We consider an attack to be stealthy if the resulting energy consumption is low (or normal). We measure the energy required to conduct the PoC attacks to understand their overhead. More specifically, we quantify the energy required to use the sensors and communication hardware to conduct these attacks. We use the Android Battery Historian to get an estimate of how much battery the PoC uses over a time period. We are then able to calculate the rate at which the attacks consume the device's battery. By measuring the battery drawn from the attack, we are able to understand how a user may notice the attack. For these experiments, we used a Google Pixel 6a (Android 12) and a Google Pixel Watch (Wear OS 3.5).

Another benefit of MadCap is that it does not need to be constantly using every sensor on the device. In MadCap, we use a *coordinated* attack and only use the relevant sensors when necessary. In an *uncoordinated* attack, every sensor is always on. A coordinated attack has a **Q**uiescent mode and an **A**ttack mode. More formally, we can think of the energy used as a random variable, A for when the attack is **A**ctive, Q for when MadCap is **Q**uiescent and waiting to start the attack. The distribution for power used for Q will be similar to a device under normal operating times. The distribution for the total amount of energy used by MadCap will be a weighted average of the two individual distributions $\gamma A + (1-\gamma)Q$, where $\gamma \in [0,1]$ is the relative amount of time the attack is active. As $\gamma \to 1$, the attack is **active** more often and **uncoordinated**. As $\gamma \to 0$, the attack is **inactive** more often and **coordinated**. For PoC1, a 2020 study [49] showed the average person visits an ATM

4.4 times per week. Assuming 5 minutes per visit yields γ_1 =2.18e-3. People spend 1.95 hours a week socializing and communicating [47]. Using this as an upper bound for PoC2 we get γ_2 =1.16e-2. For PoC3, it has been shown people spend an average of 5.48 hours/week working out, which gives γ_3 =3.26e-2. Table 8 shows the power consumption in our coordinated MadCap attack vs an uncoordinated attack.

Table 8: Power consumption of each attack.

PoC#	Device	**Q**uiescent State	**A**ttack State (Uncoord.)	MadCap (coord)	γ_i	Uncoord. / MadCap
1	Mobile	0.08 %/hr	0.08 %/hr	0.08 %/hr	2.18e-3	1
1	Wearable	0 %/hr	0.31 %/hr	6.8e-4 %/hr	2.18e-3	458.7
2	Mobile	0.259 %/hr	2.814 %/hr	0.289 %/hr	1.16e-2	9.7
2	Wearable	0 %/hr	0.84 %/hr	9.7e-3 %/hr	1.16e-2	86.2
3	Mobile	0 %/hr	0.85 %/hr	2.8e-2 %/hr	3.26e-2	30.7
3	Wearable	0.16 %/hr	0.39 %/hr	0.16 %/hr	3.26e-2	2.4

* Comparison of the efficiency between an uncoordinated attack and MadCap. Note that some values are zeroes since there are no sensors required in that state.

7 Answering user privacy queries

Our goal here is to assist users in understanding the feasibility of privacy risks from a given set of apps on multiple connected devices. Consider a wearable app that *can read the heart rate* and a corresponding companion app on the smartphone that *can communicate externally through the internet*. As we show in Section 6, there are several possible privacy risks associated with these two apps. For example, the user's heart rate can be leaked to an external server and an external entity can monitor users' emotions (through the heart rate analysis [55]). On the other hand, there are certain privacy risks that are *not* possible, e.g., leaking the user's heart rate to all people in the contact list. This is because the app needs to read the contact list, which is not granted to any app.

Challenges. There are three main challenges in achieving this. *First*, we need a way to identify all the possible behaviors of individual apps (*behavior identification*). Although various program analysis techniques [6, 20, 65] exist for behavior identification, they either suffer from false positives or false negatives [4]. Furthermore, the use of native code [2] and obfuscation [18] further exacerbates the problem. *Second*, we need to filter out which of these behaviors are privacy sensitive (*filtering*). This is because users vary in their privacy consciousness [53], and consequently, filtering privacy-sensitive behavior is user-specific. *Finally*, we need to identify the combinations of all the behaviors and report them in a user-friendly description. This is a known hard problem [3, 70], mainly because users' perspectives might differ based on the phrasing of the report. For instance, the report "reads storage and communicates through the internet" seems less alarming than "leaks your data to an external entity on the internet".

Intuition. Our intuition is that permissions' capabilities (or behaviors) can be modeled as an action (or verb) and related subjects (or nouns). For instance, one of the behaviors allowed by `READ_SMS` is the "ability to read users SMS messages", here the action is "read" and the subject is "SMS messages". User queries can also be modeled in the same way. Given a query (i.e., action on a subject) and a list of permissions, we try to match if the action in the query is allowed by the permissions.

7.1 PerQry

We implement our approach in a tool named PerQry. It consist of a one time *Analysis* phase and a continuous *Interaction* phase.

Analysis. Here, first, the user provides a list of apps (APKs), from which we extract the list of permissions by parsing `AndroidManifest.xml` of each of the app. Alternatively, the user can directly provide the list of permissions. e.g., `INTERNET` and `BODY_SENSORS`. Next, for each of the permissions, we extract its description from the official Android documentation [2]. We then tag verbs and nouns in each of these descriptions using Part of Speech (POS) tagging. The analysis is a one-time phase for a given set of permissions.

Interaction. After the analysis, the user can interact by asking all queries of interest in natural language. For every query, we perform POS tagging to identify verbs and nouns. Then, we try to find these verbs (actions) and nouns (subjects) in the POS tags identified in the analysis phase. To handle related words, we use synonyms-based matching, i.e., two words are matched if they are exactly the same or synonyms. e.g., "leaked" and "send data" are matched as they mean the same (synonyms) in the context of privacy.

We then measure the total number of actions and subjects from the user query matched with that of permissions. If the matched percentage is more than a configurable *threshold*, we consider the behavior described by the query as feasible, else not. To assist further research, we made PerQry available online and can be accessed through a browser [51].

7.1.1 Effectiveness. To evaluate the effectiveness of PerQry, we need a dataset of permission queries along with the expected answer. Unfortunately, there is no labeled dataset of privacy queries.

Dataset. To handle this, we manually created a query dataset. We create 5 categories each with 5 groups of permissions. Category 1 has items that have a single permission in each, category 2 with two permissions in each, and so on. For each of the 25 groups, we have one query. For each category, of the 5 queries, 3 are positive and 2 are negative. A positive query means the answer to the query is a "Yes" and a negative query means the answer is a "No". The mix of positive and negative queries helps us evaluate both precision and recall of our PerQry.

Examples. An example of a positive query (i.e., the expected result is "Yes, it is possible") for a permission group with two permissions (`BODY_SENSORS`, `ACCESS_FINE_LOCATION`) is *"Can the app leak where I work out?"*. Here as we can see that `BODY_SENSORS` permission enables the app to read heart rate, which can be used to infer workout activity. Similarly, location permission enables the app to precisely find the location. These two information pieces can be combined to know "where" (i.e., location) the corresponding user works out. On the other hand, a negative query, for a permission group with one permission (`BODY_SENSORS`) is *"Can the app leak my contacts?"*. Here the above permission only enables apps to read sensor information and does not provide access to contacts. Hence, the expected outcome of the above query is "No".

Results. Table 9 shows the result of running PerQry on our dataset by setting the matching threshold to 70%, as it gives the maximum overall accuracy. As mentioned in Section 7.1, *threshold* indicates the percentage of nouns and verbs in the given query that should

[2] https://github.com/aosp-mirror/platform_frameworks_base

Table 9: Effectiveness of PerQry with threshold 70%.

Cat.	# permissions per group	# queries		TP Rate	FP Rate	Precision	Recall
		Positive	Negative				
1	1	3	2	1	0.5	0.75	1
2	2	3	2	1	0.5	0.75	1
3	3	3	2	1	1	0.6	1
4	4	3	2	0.66	0	1	0.66
5	5	3	2	0.33	0	1	0.33
Cumulative		15	10	0.6	0.3	0.75	0.75

* As mentioned in Section 7.1.1, each category contains 5 groups of permissions. The number of permissions in a group varies with different permissions. There is one query (positive or negative) per each permission group.

be matched in the permissions description. The overall precision and recall of PerQry is 75% on our dataset.

Finding 5: It is feasible to answer users' multi-device privacy-related questions using NLP techniques on permissions.

7.1.2 Limitations. Although we aim to answer all user queries precisely, the tool currently cannot handle "non-first-order queries". PerQry tries to answer user queries under the assumption that the queried behavior is present directly in the behaviors extracted from the apps' permissions. We call such queries first-order as there is a direct correspondence between queried behavior and apps' behaviors. However, there are certain behaviors that cannot be identified directly from the apps behavior as they require additional reasoning over apps behaviors. For instance, as shown by the recent study [55], we can detect user emotions using `INTERNET` and `BODY_SENSORS` permissions. However, according to PerQry, this is not possible because additional reasoning of *mapping human emotions to heart rate* is missing in the behaviors extracted from apps permissions. We call such queries *non-first order queries*. These cannot be answered by PerQry.

7.2 Integration

Here, we discuss how PerQry can be integrated into the mobile OS to improve the security and privacy of users' devices.

Play Store service. The current mechanism adopted by most mobile OSes for capturing malware apps focuses their effort on the app submission process. Here, developers must follow market guidelines and pass many filters to be able to publish an application in the app market. However, as it has become evident, this process has faults and does not constitute the best way to detect malicious applications [57, 67]. A malware app may bypass the Google Play Store security mechanism and attack thousands or even millions of victims before Google takes it down. We propose to enhance this process by integrating tools such as PerQry. In the proposed scheme, the market app (i.e., Google Play Store) has a more active role in detecting possible security infringement, besides making the user more aware of permissions requested by apps she intends to install on their devices. For example, the market app can alert the user of unusual permission requests by an app or possible collusion attack from a set of apps, including apps already installed and the app intended to be installed by the user. In the latter scenario, PerQry can analyze the union set of permissions from apps that can communicate with each other—*communication partners can be detected through static analysis techniques.* To this end, we build a prototype application that mimics the Google Play Store app's functionality, called GPlay Secure. In contrast to the default market app, our prototype warns the user of possible privacy vulnerabilities on her devices. For example, GPlay Secure alerts when an app requests sensitive permissions (e.g., `BODY_SENSOR`), or when a collusion attack is plausible.

Evaluation. We conduct a user study in AMT to assess the effectiveness of GPlay Secure minimizing privacy attacks on multi-platform scenarios. We deployed GPlay Secure on an Android emulator and crafted ten independent scenarios for the study. In each scenario, we asked the participants to install two different apps on two paired devices (i.e., a mobile and a wearable)—one app per device. Our dataset included malware apps that can collude with each other to orchestrate a privacy attack. We divided the participants into two groups. The first group (experimental) interacted with GPlay Secure to install the apps. In contrast, the second group (control) used a market app that did not include our modifications. This app's functionality mimicked the Google Play Store and was used to validate if GPlay Secure provides any improvements to the default app. We had 120 participants in the study but removed those that failed the control questions for quality purposes, leaving us with 51 and 52 in the experimental and control group, respectively (103 total).

Results. Overall, our proposed solution was able to reduce the number of malicious apps installed by 8% (381 in the control group vs. 352 in the experimental group). Moreover, GPlay Secure was not only useful to reduce malwares, but also to increment the engagement of the participants in 20s per scenario with respect to the participant from the control group.

Our technique, GPlay Secure, dissuaded 20 of the 51 participants (on 45 total occasions) from installing a malicious app on the devices, which means that participants acknowledged the possible risk and decided to install another app.

8 Related Work

Privacy attack on Android and contextual inferencing. Privacy attacks on Android phones is a widely explored area, including privacy threats due to location sharing [14], sharing of crowdsensed or crowdsourced information [15], or sharing of other context, such as which apps are being used together [13]. The defense mechanism spans techniques from using randomization, adding noise *a la* differential privacy [40], applying holomorphic identity-based encryption (IBE) [25], and many others. However, these works fail to consider using information from multiple devices belonging to the same user. Zimmeck *et al.* [72] puts together information for a specific user across devices but focuses on vastly different contexts involving browsing websites. More directly, our work it is inspired on early research that leverages *context* for designing accurate inferencing on mobile phones or for mobile crowdsensing [46, 69]. Our attack work turns this on its head and uses context, from two devices, to expose the privacy attack surface. Some of this context-based work has also been done for wearables [26]. As noted earlier, we borrow this idea of inferring context and then apply it to decide the most opportune time window for a privacy attack.

Privacy attack on wearables. An early seminal work that sounded the alarm on possible privacy breaches due to the use of wearable sensors was by Raij *et al.* [52]. Several other works have shed more light on the privacy implications of widespread wearable usage [16, 17, 73]. A common pattern is called the "input inference attack", which consists of two steps: keystroke detection and keystroke inference [38, 44, 62]. Keystroke detection can also be performed by utilizing the audio signal [38] recorded by the microphone available on the smartwatch. Our work can be thought of as a logical extension of this line of work. We infer the context from multiple devices related to user's physical signals (e.g., heart rate) from the wearable and cyber signals (e.g., location) from the smartphone. Further, so far there has been a separation between two domains. Some take a human-centric view of the privacy issues (and perform user surveys and/or qualitative interviews) and a disjoint set of works take a technology-centric view outlining attacks and defenses. Our work bridges the two and uses the survey and app executed by the users to understand the attack surface and then to design practical attacks on real devices.

NLP for privacy. It is known that privacy policies are hard to understand for common users [53]. Therefore, automated privacy policy creators [3, 70] have been proposed and shown to be effective. Nan *et al.*[45] used keyword matching to identify privacy-sensitive APIs. Our work focuses on custom queries based on the given data. Unlike prior works, we have neither a known set of keywords (e.g., privacy-related keywords) nor a known format of the questions. Extensive work on using NLP techniques to analyze app reviews [32, 59] use supervised learning techniques and require a large amount of labeled data. Unfortunately, there is no dataset for first-order privacy queries, which makes the existing techniques inapplicable.

9 Conclusion

We have raised the issue of user-centric privacy in the context of multiple personal devices, e.g., mobiles, wearables, and smart appliances. The tight coupling between these devices raises a new attack surface for privacy violations. Further, context as gathered from one or more devices can make these attacks more stealthy. Our user study through a survey, a permission discovery tool Permeso, and an NLP query tool for privacy scenarios PerQry provide three key insights. First, a significant majority of the users have no qualms about granting permissive permissions spanning multiple devices (Table 1). Second, there is a significant discrepancy between the survey responses on what permissions the user feels safe providing and the actual permissions they have provided on their wearable and mobile devices (Table 4). This discrepancy is more pronounced for some categories of apps, such as *Music & Audio* and *Productivity*. We find that counter-intuitively, this discrepancy is *not* dependent on the popularity of the apps (Figure 1), i.e., it is *not* the case that the user is granting more permissions to popular apps relative to what she says in the survey, compared to apps from obscure sources (within each application category). Third, looking at the top 193 wearable apps, we discover that even when permissions given by a single app are not enough to enable many of the multi-device privacy attacks, multiple apps can collude to achieve many of the attacks (Figure 3 and Table 7). Finally, our proposed solution based on the integration of PerQry to the Google Play Store services, provides a better barrier to catch malicious coordination between apps; thus enhancing user privacy.

References

[1] Fauzia Idrees Abro, Muttukrishnan Rajarajan, Thomas M Chen, and Yogachandran Rahulamathavan. 2017. Android Application Collusion Demystified. In *Future Network Systems and Security*, Robin Doss, Selwyn Piramuthu, and Wei Zhou (Eds.). Springer International Publishing, Cham, 176–187.

[2] Vitor Afonso, Antonio Bianchi, Yanick Fratantonio, Adam Doupé, Mario Polino, Paulo de Geus, Christopher Kruegel, and Giovanni Vigna. 2016. Going Native: Using a Large-Scale Analysis of Android Apps to Create a Practical Native-Code Sandboxing Policy. In *NDSS* (San Diego, CA, USA). doi:10.14722/ndss.2016.23384

[3] Manar Alohaly and Hassan Takabi. 2016. Better privacy indicators: a new approach to quantification of privacy policies. In *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*. USENIX Association, Denver, CO, USA.

[4] Amit Seal Ami, Kaushal Kafle, Kevin Moran, Adwait Nadkarni, and Denys Poshyvanyk. 2021. Systematic mutation-based evaluation of the soundness of security-focused android static analysis techniques. *ACM Transactions on Privacy and Security (TOPS)* 24, 3 (2021), 1–37.

[5] AppBrain. 2024. Most popular Google Play categories. https://www.appbrain.com/stats/android-market-app-categories.

[6] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. 2014. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *ACM Sigplan Notices* 49, 6 (2014), 259–269.

[7] American Heart Association. 2024. Target Heart Rates Chart. https://www.heart.org/en/healthy-living/fitness/fitness-basics/target-heart-rates.

[8] Edgardo Barsallo Yi, Heng Zhang, Amiya K. Maji, Kefan Xu, and Saurabh Bagchi. 2020. Vulcan: Lessons on Reliability of Wearables through State-Aware Fuzzing. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services* (Toronto, Ontario, Canada) *(MobiSys '20)*. ACM, New York, NY, USA, 391–403. doi:10.1145/3386901.3388916

[9] Vincent Bindschaedler and Reza Shokri. 2016. Synthesizing Plausible Privacy-Preserving Location Traces. In *2016 IEEE Symposium on Security and Privacy (SP)* (San Jose, CA, USA). IEEE, IEEE Computer Society, 546–563. doi:10.1109/SP.2016.39

[10] Spyros Boukoros, Mathias Humbert, Stefan Katzenbeisser, and Carmela Troncoso. 2019. On (the lack of) location privacy in crowdsourcing applications. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1859–1876.

[11] Kunlin Cai, Jinghuai Zhang, Zhiqing Hong, William Shand, Guang Wang, Desheng Zhang, Jianfeng Chi, and Yuan Tian. 2024. Where Have You Been? A Study of Privacy Risk for Point-of-Interest Recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. ACM, New York, NY, USA, 175–186. doi:10.1145/3637528.3671758

[12] Supriyo Chakraborty, Chenguang Shen, Kasturi Rangan Raghavan, Yasser Shoukry, Matt Millar, and Mani Srivastava. 2014. ipShield: A Framework For Enforcing Context-Aware Privacy. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. USENIX Association, Seattle, WA, 143–156.

[13] Saksham Chitkara, Nishad Gothoskar, Suhas Harish, Jason I. Hong, and Yuvraj Agarwal. 2017. Does this App Really Need My Location? Context-Aware Privacy Management for Smartphones. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 42 (Sept. 2017), 22 pages. doi:10.1145/3132029

[14] Yohan Chon, Nicholas D Lane, Yunjong Kim, Feng Zhao, and Hojung Cha. 2013. Understanding the coverage and scalability of place-centric crowdsensing. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Zurich, Switzerland) *(UbiComp '13)*. ACM, New York, NY, USA, 3–12. doi:10.1145/2493432.2493498

[15] Delphine Christin. 2016. Privacy in mobile participatory sensing: Current trends and future challenges. *Journal of Systems and Software* 116 (2016), 57–68.

[16] Karel Dhondt, Victor Le Pochat, Yana Dimova, Wouter Joosen, and Stijn Volckaert. 2024. Swipe Left for Identity Theft: An Analysis of User Data Privacy Risks on Location-based Dating Apps. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, Philadelphia, PA, 5053–5070.

[17] Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. 2017. Is the data on your wearable device secure? An Android Wear smartwatch case study. *Software: Practice and Experience* 47, 3 (2017), 391–403.

[18] Shuaike Dong, Menghao Li, Wenrui Diao, Xiangyu Liu, Jian Liu, Zhou Li, Fenghao Xu, Kai Chen, Xiaofeng Wang, and Kehuan Zhang. 2018. Understanding Android Obfuscation Techniques: A Large-Scale Investigation in the Wild. In *International Conference on Security and Privacy in Communication Systems*. Springer, Springer International Publishing, Cham, 172–192.

[19] Charalampos Doukas, Luca Capra, Fabio Antonelli, Erinda Jaupaj, Andrei Tamilin, and Iacopo Carreras. 2015. Providing generic support for IoT and M2M for mobile devices. In *The 2015 IEEE RIVF International Conference on Computing Communication Technologies - Research, Innovation, and Vision for Future (RIVF)*. IEEE, 192–197.

[20] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. 2014. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)* 32, 2 (2014), 1–29.

[21] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. 2011. Android permissions demystified. In *Proceedings of the 18th ACM Conference on Computer and Communications Security* (Chicago, Illinois, USA) *(CCS '11)*. ACM, New York, NY, USA, 627–638. doi:10.1145/2046707.2046779

[22] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. 2012. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security* (Washington, D.C.) *(SOUPS '12)*. ACM, New York, NY, USA, Article 3, 14 pages. doi:10.1145/2335356.2335360

[23] Marco Furini, Silvia Mirri, Manuela Montangero, and Catia Prandi. 2019. Privacy perception and user behavior in the mobile ecosystem. In *Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good* (Valencia, Spain) *(GoodTechs '19)*. ACM, New York, NY, USA, 177–182. doi:10.1145/3342428.3342690

[24] Sandra Gabriele and Sonia Chiasson. 2020. Understanding Fitness Tracker Users' Security and Privacy Knowledge, Attitudes and Behaviours. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. ACM, New York, NY, USA, 1–12. doi:10.1145/3313831.3376651

[25] Felix Günther, Mark Manulis, and Andreas Peter. 2014. Privacy-Enhanced Participatory Sensing with Collusion Resistance and Data Aggregation. In *Cryptology and Network Security*. Springer International Publishing, Cham, 321–336.

[26] Haodong Guo, Ling Chen, Liangying Peng, and Gencai Chen. 2016. Wearable sensor based multimodal human activity recognition exploiting the diversity of classifier ensemble. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Heidelberg, Germany) *(UbiComp '16)*. ACM, New York, NY, USA, 1112–1123. doi:10.1145/2971648.2971708

[27] Hadi Harb and Liming Chen. 2003. Gender identification using a general audio classifier. In *2003 International Conference on Multimedia and Expo. ICME'03. Proceedings (Cat. No. 03TH8698)*, Vol. 2. IEEE, II–733. doi:10.1109/ICME.2003.1221721

[28] Ross Harper and Joshua Southern. 2019. End-To-End Prediction of Emotion From Heartbeat Data Collected by a Consumer Fitness Tracker. In *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, 1–7.

[29] Wajih Ul Hassan, Saad Hussain, and Adam Bates. 2018. Analysis of Privacy Protections in Fitness Tracking Social Networks-or-You can run, but can you hide?. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 497–512.

[30] Jeremy Hsu. 2018. The Strava Heat Map and the End of Secrets. https://www.wired.com/story/strava-heat-map-military-bases-fitness-trackers-privacy/.

[31] Jim Huang. 2012. Android IPC mechanism. *Southern Taiwan University of Technology* (2012).

[32] Abdul Karim, Azhari Azhari, Samir Brahim Belhaouri, Ali Adil Qureshi, and Maqsood Ahmad. 2020. Methodology for Analyzing the Traditional Algorithms Performance of User Reviews Using Machine Learning Techniques. *Algorithms* 13, 8 (2020), 202.

[33] Nianyu Li, Christos Tsigkanos, Zhi Jin, Schahram Dustdar, Zhenjiang Hu, and Carlo Ghezzi. 2019. Poet: Privacy on the edge with bidirectional data transformations. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 1–10. doi:10.1109/PERCOM.2019.8767395

[34] Robert LiKamWa, Yunxin Liu, Nicholas D Lane, and Lin Zhong. 2013. Moodscope: Building a mood sensor from smartphone usage patterns. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services* (Taipei, Taiwan) *(MobiSys '13)*. ACM, New York, NY, USA, 389–402. doi:10.1145/2462456.2464449

[35] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. 2017. DEEProtect: Enabling Inference-based Access Control on Mobile Sensing Applications. *CoRR* abs/1702.06159 (2017). arXiv:1702.06159 http://arxiv.org/abs/1702.06159

[36] Fannie Liu, Mario Esparza, Maria Pavlovskaia, Geoff Kaufman, Laura Dabbish, and Andrés Monroy-Hernández. 2019. Animo: Sharing Biosignals on a Smartwatch for Lightweight Social Connection. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 18 (March 2019), 19 pages. doi:10.1145/3314405

[37] Renju Liu and Felix Xiaozhu Lin. 2016. Understanding the Characteristics of Android Wear OS. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services* (Singapore, Singapore) *(MobiSys '16)*. ACM, New York, NY, USA, 151–164. doi:10.1145/2906388.2906398

[38] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. 2015. When Good Becomes Evil: Keystroke Inference with Smartwatch. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15)*. ACM, New York, NY, USA, 1273–1285. doi:10.1145/2810103.2813668

[39] Benjamin Livshits and Jaeyeon Jung. 2013. Automatic mediation of privacy-sensitive resource access in smartphone applications. In *22nd USENIX Security Symposium (USENIX Security 13)*. USENIX Association, Washington, DC, 113–130.

[40] Chenglin Miao, Wenjun Jiang, Lu Su, Yaliang Li, Suxin Guo, Zhan Qin, Houping Xiao, Jing Gao, and Kui Ren. 2015. Cloud-Enabled Privacy-Preserving Truth Discovery in Crowd Sensing Systems. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15)*. ACM, New York, NY, USA, 183–196. doi:10.1145/2809695.2809719

[41] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. 2015. Powerspy: Location tracking using mobile device power analysis. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, DC, 785–800.

[42] Kristopher Micinski, Daniel Votipka, Rock Stevens, Nikolaos Kofinas, Michelle L. Mazurek, and Jeffrey S. Foster. 2017. User Interactions and Permission Use on Android. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 362–373. doi:10.1145/3025453.3025706

[43] Hooman Mohajeri Moghaddam, Gunes Acar, Ben Burgess, Arunesh Mathur, Danny Yuxing Huang, Nick Feamster, Edward W. Felten, Prateek Mittal, and Arvind Narayanan. 2019. Watching You Watch: The Tracking Ecosystem of Over-the-Top TV Streaming Devices. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. ACM, New York, NY, USA, 131–147. doi:10.1145/3319535.3354198

[44] John V Monaco. 2018. Sok: Keylogging side channels. In *2018 IEEE Symposium on Security and Privacy (SP)*. 211–228. doi:10.1109/SP.2018.00026

[45] Yuhong Nan, Zhemin Yang, Xiaofeng Wang, Yuan Zhang, Donglai Zhu, and Min Yang. 2018. Finding Clues for Your Secrets: Semantics-Driven, Learning-Based Privacy Discovery in Mobile Apps. In *NDSS* (San Diego, CA, USA).

[46] Suman Nath. 2012. ACE: exploiting correlation for energy-efficient and continuous context sensing. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services* (Low Wood Bay, Lake District, UK) *(MobiSys '12)*. ACM, New York, NY, USA, 29–42. doi:10.1145/2307636.2307640

[47] US Bureau of Labor Statistics. 2023. Time spent in leisure and sports activities, 2022. https://www.bls.gov/opub/ted/2023/time-spent-in-leisure-and-sports-activities-2022.htm. U.S. Department of Labor, The Economics Daily.

[48] Katarzyna Olejnik, Italo Dacosta, Joana Soares Machado, Kévin Huguenin, Mohammad Emtiyaz Khan, and Jean-Pierre Hubaux. 2017. Smarper: Context-aware and automatic runtime-permissions for mobile devices. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1058–1076. doi:10.1109/SP.2017.25

[49] Ken Paterson. 2020. ATM Usage and Preferences. https://javelinstrategy.com/research/north-american-paymentsinsights-us-data-summary-report-atm-usage-and-preferences.

[50] Anthony Peruma, Jeffrey Palmerino, and Daniel E Krutz. 2018. Investigating user perception and comprehension of android permission models. In *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems* (Gothenburg, Sweden) *(MOBILESoft '18)*. ACM, New York, NY, USA, 56–66. doi:10.1145/3197231.3197246

[51] Privacy Query Website. 2025. https://sites.google.com/view/madcap-privacy-attack/home.

[52] Andrew Raij, Animikh Ghosh, Santosh Kumar, and Mani Srivastava. 2011. Privacy risks emerging from the adoption of innocuous wearable sensors in the mobile environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. ACM, New York, NY, USA, 11–20. doi:10.1145/1978942.1978945

[53] Joel R Reidenberg, Travis Breaux, Lorrie Faith Cranor, Brian French, Amanda Grannis, James T Graves, Fei Liu, Aleecia McDonald, Thomas B Norton, and Rohan Ramanath. 2015. Disagreeable privacy policies: Mismatches between meaning and users' understanding. *Berkeley Tech. LJ* 30 (2015), 39.

[54] Yiran Shen, Bowen Du, Weitao Xu, Chengwen Luo, Bo Wei, Lizhen Cui, and Hongkai Wen. 2020. Securing cyber-physical social interactions on wrist-worn devices. *ACM Transactions on Sensor Networks (TOSN)* 16, 2 (2020), 1–22.

[55] Lin Shu, Yang Yu, Wenzhuo Chen, Haoqiang Hua, Qin Li, Jianxiu Jin, and Xiangmin Xu. 2020. Wearable emotion recognition using heart rate data from a smart bracelet. *Sensors* 20, 3 (2020), 718.

[56] Marcos Tileria, Jorge Blasco, and Guillermo Suarez-Tangil. 2020. WearFlow: Expanding Information Flow Analysis To Companion Apps in Wear OS. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. USENIX Association, San Sebastian, 63–75.

[57] Alanna Titterington. 2023. Google Play malware clocks up more than 600 million downloads in 2023. https://www.kaspersky.com/blog/malware-in-google-play-2023/49579/.

[58] Lynn Tsai, Primal Wijesekera, Joel Reardon, Irwin Reyes, Serge Egelman, David Wagner, Nathan Good, and Jung-Wei Chen. 2017. Turtle Guard: Helping Android Users Apply Contextual Privacy Preferences. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. USENIX Association, Santa Clara, CA, 145–162.

[59] Phong Minh Vu, Hung Viet Pham, Tam The Nguyen, and Tung Thanh Nguyen. 2015. Tool support for analyzing mobile app reviews. In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 789–794. doi:10.1109/ASE.2015.101

[60] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. 2016. Friend or Foe? Your Wearable Devices Reveal Your Personal PIN. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (Xi'an, China) *(ASIA CCS '16)*. ACM, New York, NY, USA, 189–200. doi:10.1145/2897845.2897847

[61] Chen Wang, Jian Liu, Xiaonan Guo, Yan Wang, and Yingying Chen. 2019. WristSpy: Snooping passcodes in mobile payment using wrist-worn wearables. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2071–2079. doi:10.1109/INFOCOM.2019.8737633

[62] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. 2015. Mole: Motion leaks through smartwatch sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (Paris, France) *(MobiCom '15)*. ACM, New York, NY, USA, 155–166. doi:10.1145/2789168.2790121

[63] Jice Wang and Hongqi Wu. 2018. Android Inter-App Communication Threats, Solutions, and Challenges. https://arxiv.org/abs/1803.05039

[64] Xingfeng Wang and Heecheol Kim. 2015. Detecting User Activities with the Accelerometer on Android Smartphones. *Journal of Multimedia Information System* 2, 2 (2015), 207–214.

[65] Fengguo Wei, Xingwei Lin, Xinming Ou, Ting Chen, and Xiaosong Zhang. 2018. JN-SAF: Precise and Efficient NDK/JNI-aware Inter-language Static Analysis Framework for Security Vetting of Android Applications with Native Code. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. ACM, New York, NY, USA, 1137–1150. doi:10.1145/3243734.3243835

[66] Primal Wijesekera, Joel Reardon, Irwin Reyes, Lynn Tsai, Jung-Wei Chen, Nathan Good, David Wagner, Konstantin Beznosov, and Serge Egelman. 2018. Contextualizing Privacy Decisions for Better Prediction (and Protection). In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, 1–13. doi:10.1145/3173574.3173842

[67] Davey Winder. 2024. New Google Play Store Warning—200 Dangerous Apps, 8 Million Installs. https://www.forbes.com/sites/daveywinder/2024/10/15/new-google-play-store-warning-200-dangerous-apps-800-million-installs/.

[68] Chugui Xu, Ju Ren, Liang She, Yaoxue Zhang, Zhan Qin, and Kui Ren. 2019. EdgeSanitizer: Locally Differentially Private Deep Inference at the Edge for Mobile Data Analytics. *IEEE Internet of Things Journal* 6, 3 (2019), 5140–5151.

[69] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 351–360. doi:10.1145/3038912.3052577

[70] Le Yu, Tao Zhang, Xiapu Luo, and Lei Xue. 2015. AutoPPG: Towards Automatic Generation of Privacy Policy for Android Applications. In *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices* (Denver, Colorado, USA) *(SPSM '15)*. ACM, New York, NY, USA, 39–50. doi:10.1145/2808117.2808125

[71] Jianxin Zhao, Richard Mortier, Jon Crowcroft, and Liang Wang. 2018. Privacy-Preserving Machine Learning Based Data Analytics on Edge Devices. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society* (New Orleans, LA, USA) *(AIES '18)*. ACM, New York, NY, USA, 341–346. doi:10.1145/3278721.3278778

[72] Sebastian Zimmeck, Jie S Li, Hyungtae Kim, Steven M Bellovin, and Tony Jebara. 2017. A Privacy Analysis of Cross-device Tracking. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1391–1408.

[73] Noé Zufferey, Mathias Humbert, Romain Tavenard, and Kévin Huguenin. 2023. Watch your Watch: Inferring Personality Traits from Wearable Activity Trackers. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 193–210.