When It's Better to Ask Forgiveness than Get Permission: Attribution Mechanisms for Smartphone Resources

Christopher Thompson, Maritza Johnson,
Serge Egelman, David Wagner, and Jennifer King
{cthompson, maritzaj, egelman, daw}@cs.berkeley.edu, jenking@ischool.berkeley.edu
University of California, Berkeley

ABSTRACT

Smartphone applications pose interesting security problems because the same resources they use to enhance the user experience may also be used in ways that users might find objectionable. We performed a set of experiments to study whether attribution mechanisms could help users understand how smartphone applications access device resources. First, we performed an online survey and found that, as attribution mechanisms have become available on the Android platform, users notice and use them. Second, we designed new attribution mechanisms; a qualitative experiment suggested that our proposed mechanisms are intuitive to understand. Finally, we performed a laboratory experiment in which we simulated application misbehaviors to observe whether users equipped with our attribution mechanisms were able to identify the offending applications. Our results show that, for users who notice application misbehaviors, these attribution mechanisms are significantly more effective than the status quo.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces; D.4.6 [**Software**]: Security and Protection

General Terms

Human Factors, Design, Experimentation, Security

1. INTRODUCTION

It's easier to ask forgiveness than it is to get permission.
—Grace Hopper

The top four mobile platforms by market share represent 99% of the U.S. market: Android (54%), iOS (34%), BlackBerry (8%), and Windows Phone (3%) [19]. All of these platforms enforce permission models where the user is explicitly prompted before an application is given access to personal data or hardware features. The goal of these permission models is to give users control over how applications access their devices and personal information.

Recent research has demonstrated that current permission models have serious limitations: few users read permission requests and even fewer understand them [17, 22]. In Android, the most popular mobile platform, users see an average of four permission requests each time they install an application [16]. Some permission requests are so pervasive and represent such minimal risks that their presence detracts from more concerning permission requests, thereby causing habituation [4]. Other permission requests lack context because they are presented during installation, rather than when the resource is actually needed by the application. For instance, when an application is granted the ability to collect GPS data, the application has just as much ability to collect this data when the user is at a shopping mall as when she is visiting a paramour.

After examining users' perceptions of smartphone-related risks [15], Felt et al. suggested that smartphones should not prompt the user about a permission request when the risks from that request can either be undone or when they are only an annoyance [14]. Instead, to minimize habituation, these permissions should be "implicitly granted." For this to be a viable solution, users must be able to attribute the use of an implicitly-granted permission to a specific application. Such a feature would allow the user to evaluate the appropriateness of the action in context and provide the user with an avenue for identifying an offending application so that implicitly-granted permissions may be revoked and their associated misbehaviors stopped or undone.

We performed a series of experiments on the Android platform to evaluate two different attribution mechanisms for implicitly-granted permissions. We examined interactive notification icons that identified the application that was vibrating the smartphone, as well as annotations to the Settings application that identified the application that last changed the wallpaper. Overall, we observed that when compared to the status quo (i.e., a stock Android device), once a misbehavior was noticed, significantly more participants who used devices equipped with our new attribution mechanisms were able to correctly identify the applications that caused those misbehaviors. Our contributions are as follows:

- We show that as few as 17% of smartphone users understand that background applications may have the same abilities as foreground applications, thus making it difficult to identify the source of a misbehavior.

Symposium on Usable Privacy and Security (SOUPS) 2013, July 24–26, 2013, Newcastle, UK.

- We show that as many as 85% of current Android users understand how to use the Settings application to attribute application behaviors. We observed that 95% of our participants understood that the notification drawer could be expanded to yield more information about application behaviors.
- We show that when presented with our attribution mechanisms, significantly more participants correctly identified the source of a misbehavior.

2. BACKGROUND

The literature on hazard control provides a hierarchy of procedures for dealing with potential hazards [24]: (1) eliminate the hazard; (2) guard against the hazard; otherwise, (3) warn about the hazard. Thus, warning is a last resort, whenever a hazard cannot be eliminated or otherwise mitigated. With regard to mobile devices (e.g., smartphones and tablets), hazards occur when applications inappropriately access personal data or hardware resources. Malware applications do this intentionally for nefarious purposes and therefore platforms should automatically eliminate or otherwise mitigate these threats.

In mobile security, much work has been done examining mobile malware [13, 20], malware detection [6, 7, 26], and privilege escalation [9, 18, 30]. However, recent work by Lever et al. has shown that, at least in the US, only a tiny fraction of mobile devices seem to have malware problems [25]. Instead, a more pervasive threat appears to be from well-meaning applications that—either intentionally or unintentionally—access users' personal data and/or hardware resources in ways that they neither expect nor condone.

Enck et al. showed that there is systematic misuse of personal and phone identifiers stored on Android smartphones [12]. Researchers have developed techniques to detect privacy-violating information flows [11], and to detect when applications are overprivileged [16]. Because users may evaluate these risks differently (i.e., whether the functionality offered by a particular application is worth the cost of giving up personal data), platforms warn against this hazard through the use of permission requests.

When choosing how to represent permission requests, platform developers have essentially four mechanisms at their disposal: attribution mechanisms for automatically granted permissions, trusted UI components, runtime warnings, and install-time warnings [14]. Android uses install-time warnings for all of its permission requests.

Felt et al. found that the vast majority of Android users do not look at permissions during installation. A plurality of their laboratory participants were unaware of permissions' existence, and very few could explain specific permissions when asked [17]. These results indicate that the Android permission model leaves room for improvement.

Some mobile platforms use runtime warnings (e.g., Windows Phone and BlackBerry) or eschew install-time warnings altogether (e.g., iOS). However, a decade of usable security research suggests that habituation may render these mechanisms ineffective [3, 5, 10, 29, 31]. Because habituation increases with each exposure to a warning [23], a simple way of minimizing habituation is to limit the number of warnings to which users are regularly exposed.

As another alternative, several permissions systems have been built around the concept of "trusted UI" [21, 27, 28]. In these systems, users grant applications special privileges as part of their natural flow, so their primary task is not interrupted. When a permission is requested, the user interacts with a system-drawn component, which grants the application access to the resource. However, this approach is not applicable to every possible permission request (e.g., some requests may need to be granted at a future time or otherwise cannot be integrated into the user's flow).

Based on the observation that no single approach is appropriate for all situations, Felt et al. developed a framework for choosing the most appropriate permission-granting mechanism for each given permission [14]. In order to minimize habituation, the framework recommends that users should not be explicitly prompted when applications request permission to perform actions that can either be completely undone or pose no more harm than an annoyance. Instead, these permissions should be "implicitly granted"—but in these cases, users should be provided with attribution mechanisms so that they can identify how applications are using their mobile devices and stop or undo any undesirable actions. Felt et al. estimate that 55% of Android permissions can be implicitly granted [14]. However, to our knowledge, no one has empirically evaluated the mechanisms that might be used to attribute behaviors to these implicitly-granted permissions.

3. FOUNDATIONS

In preparation for designing and evaluating our attribution mechanisms, we conducted an online survey of 189 smartphone users and performed a qualitative experiment with eight Android users.

We designed our online survey with the goal of determining whether smartphone users understand the attribution mechanisms they already have on their devices, whether they make use of those mechanisms, and whether they understand that background applications may be as likely to be responsible for inappropriate behaviors as foreground applications (thus necessitating the need for attribution mechanisms to disambiguate the cause of a misbehavior). We followed up this survey with a qualitative experiment where we met in-person with Android users to better understand where people look on their devices when they are attempting to identify the cause of a misbehavior.

3.1 Internet Survey

In February 2013, we recruited 189 Android and iOS users[1] to participate in a survey using Amazon's Mechanical Turk. Our respondents included more men than women (67.8% male), and ages ranged from 18–66 ($\mu = 28.3$, $\sigma = 8.2$). The survey began with a scenario:

> Imagine you received a very high phone bill due to data usage. Explain the steps you would take using your smartphone to determine which application was likely responsible.

The purpose of this question was to understand the first place where smartphone users would look. Additionally, all users of Android 4.0 and later have a panel within the Settings application that displays data usage on a per-application

[1] We asked participants to provide their smartphone's make and model, which allowed us to filter out users of other platforms post-hoc. Likewise, we also included instructions for participants to find and report the exact version of their smartphone's operating system, however, we did not collect this information from 65 of our 189 respondents.

basis. Thus, we were curious if these users understood that this feature already existed on their devices. They did: of the 45 respondents who used Android 4.0 or later, 33 (73%; 95%CI: [58%, 85%]) indicated that they would access this sub-panel of the Settings application. Among our other respondents, the most common response was that they would blame whatever application they used the most (32% of 79; 95%CI: [22%, 43%]):

- *"First, I would think about which apps I use most frequently. Then, I would open those apps and try to figure out which ones use a lot of data."*
- *"I would check to see which apps are active the most."*
- *"I might first try to narrow it down to any apps most recently installed."*
- *"I would likely open my task manager to see what was running."*

Surprisingly, the next most common response among users of iOS and earlier version of Android was that they would inspect their device's Settings application for clues (21.5% of 79; 95%CI: [13%, 32%]). Despite the fact that these respondents do not actually have this feature, our results indicate that placing this attribution mechanism within the Settings application is intuitive to many smartphone users.

As of iOS 6, users now have the ability to use the Settings application to monitor how applications access location data, contacts, calendars, reminders, the photo library, and Bluetooth pairing [8]. We included a multiple choice question in the survey to examine whether the 40 respondents who used iOS 6 understood that they had this ability:

> Some smartphones allow you to monitor how apps make changes or use system resources. Which of the following resources do you believe you can monitor on your device (i.e., determine which app was responsible)?

We supplied respondents with multiple options, including four of the aforementioned iOS 6 resources (we omitted reminders and Bluetooth pairing). We included six other resources so as to not prime participants to the new iOS 6 attribution mechanisms (we therefore do not include these six others in our analysis). On average, half of our respondents who used iOS 6 were aware of these attribution mechanisms, further corroborating our earlier findings that the Settings application is an intuitive location for this type of information:

- Location: 55% of 40 (95%CI: [39%, 71%])
- Photos: 43% of 40 (95%CI: [27%, 59%])
- Contacts: 40% of 40 (95%CI: [25%, 61%])
- Calendar: 35% of 40 (95%CI: [21%, 52%])

Finally, we asked participants about background applications to determine whether smartphone users understand that an application running in the background maintains the same abilities that it had when in the foreground:

> Imagine you are using a smartphone application and then return to the home screen to launch a different app. What happens to the original app?

We provided participants with four multiple-choice answers (in addition to "other" and "I don't know"):

- *"It runs in the background, but with fewer abilities"* (32.8% of 189; 95%CI: [26%, 40%])
- *"It goes into a suspended state"* (27.5% of 189; 95%CI: [21%, 35%])
- *"It runs in the background, but with the same abilities"* (22.2% of 189; 95%CI: [17%, 29%])
- *"It stops running"* (6.9% of 189; 95%CI: [4%, 12%])

For both iOS and Android, the correct answer is that "it runs in the background, but with the same abilities." On Android, a background application can even draw over other applications. We do note that some participants may have based their answer on a background application's limited ability to capture input events. However, neither operating system limits the API calls of a background application[2]. This answer was selected by only 42 respondents (22% of 189). We did not observe statistically significant differences between users of iOS and Android. This finding by itself strongly shows the need for attribution mechanisms to help users better understand how applications are accessing a mobile device's resources, since they are otherwise likely to incorrectly attribute a misbehavior to whatever application was running in the foreground at the time that they noticed the misbehavior.

3.2 Qualitative Experiment

We followed up our online survey with an in-person qualitative experiment. Our goals were to better understand how users discover resource usage information, whether users notice passive notifications, and whether users know that notifications in the status bar can be expanded. We designed three scenarios to investigate these questions:

1. We first asked participants to use the smartphone to identify the likely cause of a data overage. The purpose of this task was to see whether they would navigate to the Settings application.
2. Next, we asked each participant to use the smartphone to determine the cause of an unexpected change to the smartphone's wallpaper. The purpose of this task was to see where participants might expect to find information to identify the cause of this change.
3. Finally, we had them perform a distraction task (i.e., use a news reader application) while we made a notification appear in the status bar. The purpose was to see whether they would notice the notification and whether they knew that the notification drawer could be expanded to show additional information.

We recruited participants from Craigslist, offering them $35 cash in exchange for their time. Our only screening requirement was that every participant had been an Android user for at least six months. Our sample consisted of eight people (5 female, 3 male), who ranged in age from 25–49.

We met each participant individually at a local coffee shop. To begin, we told the participant that the purpose of the study was to understand how people interact with

[2] While Apple's review process is supposed to identify and reject certain behaviors while an application is in the background, and a background application is more likely to be killed if it deviates from these guidelines, there are no access control mechanisms to enforce them [2].

smartphones. We gave an overview of the session and asked the participant to think aloud as he or she completed each scenario. As an example of how to think aloud while completing a task, we asked the participant to find the version of Android running on their personal device. This task served the dual purpose of revealing which version of Android they used. Three of the participants' smartphones ran Android 4.0 or later, three ran a variant of Android 2, and two participants did not have their smartphones with them.

During the experiment, the participant used a Galaxy Nexus running Android 4.2 to complete the scenarios. We created a background application which we installed on the phone to display a custom notification during the third scenario. For the first task, we asked the participant to imagine:

> On your latest phone bill, you see that you have unexpected overage charges for going over your data plan. You don't remember using your smartphone more than usual and suspect something happened on the smartphone. How do you identify what caused the overage (or what is primarily responsible)?

We analyzed the transcripts to identify whether the participant found the data usage screen in the Settings application on their own, and whether they had experience with the screen. Those who had experience with Android 4.0 or later may have seen this screen before; the screen would have been a new feature for those with an older version of Android. Half of the participants, four of eight, immediately visited the data usage screen. Two participants found the screen after explaining other avenues they would explore first, and two participants found the screen only after the experimenter prompted them to explore the smartphone. Half of the participants said they would call their service provider first, and only explored the smartphone after the experimenter's prompting. None of the participants claimed to have seen the data usage screen prior to the experiment, not even the three who owned smartphones running Android 4.0 or later.

For the second scenario, the experimenter asked:

> Imagine you notice that the wallpaper is a picture of Justin Bieber even though it used to be a picture from your last vacation. How would you change the wallpaper back to a picture of your choice? Imagine you are certain that you didn't change the background picture, how would you figure out what happened?

We designed the scenario knowing it was unsolvable—we wanted to understand where people go to discover information when an unexpected change occurs. All of the participants correctly named a way to correct the wallpaper. But, because there is no way to identify how the wallpaper changed, the steps participants took to identify the responsible application varied. Three participants said they would suspect a friend, three others said they would assume the smartphone was bumped while in a pocket or purse, and one would attribute the change to a glitch in the software. Thus, without prompting, very few stated that an application could be responsible.

After a reminder that an application could be the culprit, four participants suggested they would look for relevant information in the Settings application. One participant said he would list his recently installed applications and uninstall them. He continued to say that if he was unable to single out the offending application, he would wipe his smartphone, something he claimed to have done before.

In the third scenario, we asked each participant to browse the day's headlines using the BBC News application. After thirty seconds, a second study coordinator caused a passive icon to appear in the status bar that indicated that the Facebook application was using the camera to record.[3] We designed the experimental protocol such that the participant had one minute to interact with the notification prior to being prompted. None of the participants did so.

One minute after the icon appeared, the experimenter asked, *"Is there an icon in the upper lefthand corner of the status bar?"* At this point, every participant acknowledged the presence of the icon. When the experimenter asked, *"Did you notice the icon while you were using the BBC application?"*, only three of the participants said they did. After their attention was drawn to the icon, all of the participants except one knew what the icon meant and successfully interacted with the icon to learn that Facebook had enabled the video recorder. Half of the participants claimed to be unaware that an application could enable the video recorder from the background, but the experimenter quickly explained that the notification was fake.

As a final question, the experimenter asked, *"What should you be able to do from the notification?"* Half of the participants said they would want to know why the application was using the resource, and a link to the application would be useful. Other participants expressed that a way to stop the recording would be most useful.

3.3 Design Implications

Our online survey and qualitative experiment suggest that many Android users are unaware that applications running in the background may be able to access the same hardware and software resources that they associate with applications running in the foreground. Likewise, users do not readily associate system changes with misbehaving applications. This indicates that attribution mechanisms are needed to help users better understand how third-party applications are using their devices. At the same time, most Android users appear to be familiar with both the status bar and Settings application, and look to these for additional information.

Our results suggest that when system changes occur, it may be helpful to place attribution information in places where users might go to undo those changes. For instance, all of our qualitative experiment participants correctly indicated how to undo an undesirable wallpaper change. Thus, it may be intuitive to annotate these settings panels with information to attribute the source of the previous wallpaper change (i.e., the user or a specific application). Likewise, our results show that users are familiar with the status bar and understand that they can expand it to yield additional information about an application's behavior. Thus, we believe that this may be an intuitive location for an attribution mechanism to identify the source of an ongoing behavior

[3] In our original design, we used a download icon. However, our first participant noted that he noticed the icon, but did not interact with it because it was probably just an automatic update, which happens often. Therefore, we switched to using the video recording icon because we thought users would be more motivated to investigate it.

(e.g., naming the application that is causing the device to vibrate over a prolonged period of time). We decided to implement these two attribution mechanisms and evaluated them in the laboratory.

4. METHODOLOGY

Presently, smartphones allow users to discover only limited information about how applications utilize device resources. If an application is misbehaving, in many cases the use of a system resource might go unnoticed by the user. In our research, we sought to address these shortcomings by designing ways to communicate the use of system resources to users. In our study, we limited our investigation to permissions that protected users from relatively minor risks (i.e., annoyances or changes that could be undone), which could therefore be implicitly granted. The specific attribution mechanisms that we examined were passive notifications and settings panel annotations.

We conducted a laboratory experiment to examine how people react to applications that misbehave, with a focus on the steps they take to attribute the misbehavior. In this section, we describe the conditions we tested in our study, the tasks we asked participants to complete, and the overall protocol for a single session. We told all participants that they were recruited to review new applications, and while doing so, we triggered misbehaviors and observed whether they would use our attribution mechanisms to identify the applications responsible for the misbehaviors.

4.1 Conditions

During the study session, we provided each participant with a Samsung Nexus S running Android 4.1.1 to use during the experiment. We instrumented all of the smartphones, as described in Section 4.3. We gave participants assigned to the *Control* condition smartphones that we instrumented to misbehave and log user events. We also gave participants assigned to the *Experimental* condition instrumented smartphones, but these smartphones had additional features, including passive notifications to indicate an application's use of the vibrate permission (Figure 1), and attribution information to note changes to the wallpaper (Figure 2).

4.2 Task Design

We exposed participants to two different misbehaviors, each in two phases. The misbehaviors consisted of an application causing the smartphone to continuously vibrate and an application changing the wallpaper; the order of the misbehaviors was randomized by session. The first phase of each misbehavior occurred while participants were engaged in an application evaluation task. During this phase, we did not prompt participants about the misbehavior and observed whether or not they would notice the misbehavior, investigate it, and then comment about it in their evaluations. In the second phase, the experimenter alerted the participants to the misbehavior and prompted them to identify the responsible application. We included both phases as a result of the qualitative experiment described in Section 3.2. In that experiment, none of the participants interacted with the passive icon that appeared in the upper left hand corner of the status area without us first prompting them. Once prompted to comment on it, nearly all participants understood what the notification signified and how to interact with it to discover additional information.

When we did not prompt participants, each misbehavior was masked by an application evaluation task: we told participants that their primary task was to write reviews. For the vibration misbehavior, we installed three timer applications that had been granted the vibration permission, so that each application could be plausibly blamed for the vibration misbehavior if a participant looked at the permission manifests within the Settings application. We gave participants a list of applications to explore and then review in order. Each list contained the three timer applications in random order, followed by a dice game. While the participants explored the dice game, we made it appear as though one of the timer applications was making the phone continuously vibrate.

For the wallpaper misbehavior, we installed three drawing applications that each required permission to change the wallpaper. To increase the likelihood that participants noticed the changed wallpaper, we asked participants to evaluate three widgets after using and reviewing the three drawing applications.[4] We randomized the order in which participants first ran each of the drawing applications.

We intended for participants to be surprised by the misbehaviors, and suspected that some participants might not notice that the wallpaper had been changed. Without prompting, we hypothesized that participants in the *Control* condition would attribute the misbehaviors to the applications running in the foreground when the misbehaviors occurred.

4.3 Smartphone Instrumentation

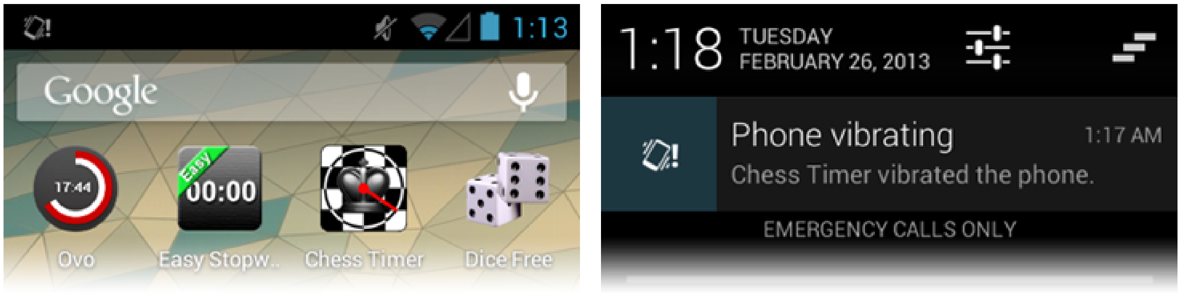
We used the Android Open Source Project (AOSP) [1] to create a modified version of Android 4.1.1, the version recommended for the Nexus S. We created forks for both the *Experimental* and *Control* conditions. We instrumented both to log when the participant opened the Display Settings, when the participant opened the Wallpaper Chooser, and when the participant opened the recent tasks menu. We used the existing Android system logs to identify when applications were started or stopped, when the user visited the home screen, when the user expanded the notification drawer to view notifications, and when the user visited the Apps area of the Settings application.

To control the misbehaviors in both conditions and the attribution information in the *Experimental* condition, we created a custom application that allowed us to trigger each misbehavior and blame a randomly-selected running application. The vibration misbehavior pulsed the phone's vibration motor twice every seven seconds, for a duration of roughly five minutes, or until the blamed application was killed. A persistent notification for the vibration was displayed on the *Experimental* smartphones (Figure 1). The wallpaper misbehavior changed the smartphone's wallpaper to a picture of Justin Bieber. We modified the *Experimental* smartphones to annotate the Display Settings and Wallpaper Chooser with provenance information (see Figure 2).

4.4 Protocol

We assigned each participant an individual desk equipped with dividers, a smartphone, and a laptop. The laptop was

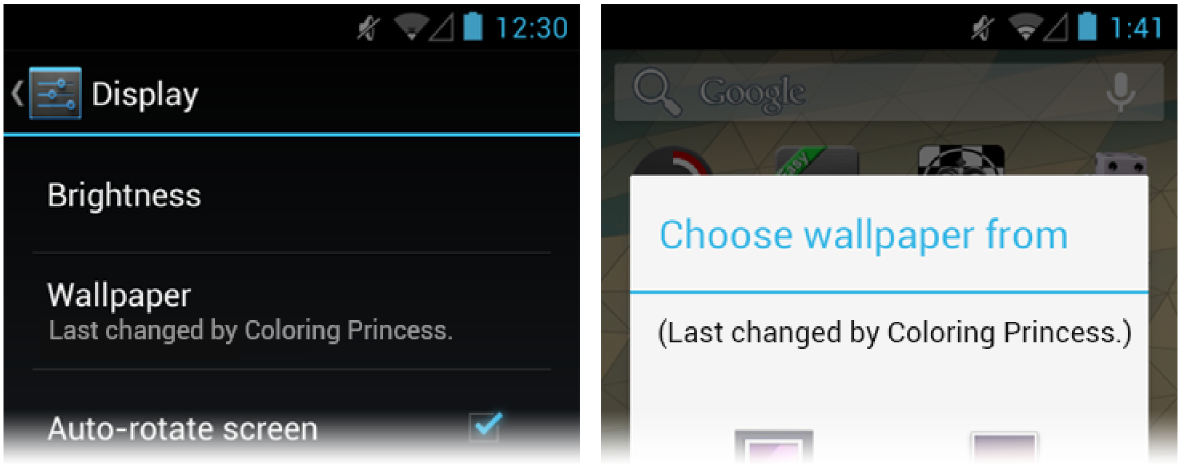
[4] Android widgets are lightweight informational applications that allow users to customize their home screens. Because they do not occlude the home screen, we reasoned that a user is more likely to notice a wallpaper change when interacting with a widget rather than a fullscreen application.



Phone vibrating 1:17 AM
Chess Timer vibrated the phone.

(a) *Vibration Notification* (b) *Expanded Vibration Notification*

Figure 1: The vibration notification icon in the status area (a) and after the user has expanded it (b).



Brightness

Wallpaper
Last changed by Coloring Princess.

Auto-rotate screen ✓

Choose wallpaper from

(Last changed by Coloring Princess.)

(a) *Annotated Settings* (b) *Annotated Chooser*

Figure 2: The wallpaper change attribution in the Display Settings (a) and in the Wallpaper Chooser (b).

used for completing an online survey that presented instructions, the randomized list of applications to review, and survey questions at check points: after each of the two evaluation periods (wherein the misbehaviors first occurred unprompted), after each of the two misbehaviors occurred with prompting, and then at the end of the experiment (i.e., the exit survey). We told participants that the primary purpose of the study was exploring and reviewing applications, and the majority of the session was spent on this task (approximately 45 minutes).

Before participants arrived at the lab, we randomly distributed the instrumented smartphones among the desks. When the participants arrived, we instructed them to sit at a desk with a smartphone and to review the consent form while they waited for the session to begin. In this manner, the assignment of the two conditions was double blind (we later determined the condition assignment by examining the logs stored on each smartphone). We ran each session according to a timed schedule, and so we could not admit latecomers. Each session was conducted by two study coordinators: one read the script and distributed the study material, while the other observed and initiated the application misbehaviors.

The instructions on the laptops provided participants with a list of applications to explore and then review. For clarity, we will present the protocol as though the vibration misbehavior occurred first. For the first task, we told participants:

> Your task is to review three timer apps and a dice game. Your task is to play with apps on the smartphone in the order listed on the laptop for three minutes each. Use the scratch paper provided to take notes about what you like and don't like about each app. You will use these notes later to write reviews about each app.

Participants spent the next twelve minutes playing with each application in the order listed on the laptops in front of them. While participants explored the dice game, after exploring all three timer applications in random order, the vibration misbehavior occurred. We included the dice game to ensure that all three timer applications were running in the background at the time of the misbehavior. Likewise, we believed participants would hold the devices to use the dice game, and therefore would be more likely to notice the vibration. The study coordinator distributed four review sheets before the exploration period ended, and asked participants to spend the next ten minutes writing a review of each application. The study coordinator announced the passage of time in two and a half minute intervals.

For the second task, we told participants:

> Your task is to review three drawing apps and some widgets. To begin, your task is to play with apps on the smartphone in the order listed on the laptop for three minutes each. Use the

> scratch paper provided to take notes about what you like and don't like about each app. You will use these notes later to write reviews about each app.

Again, participants spent twelve minutes exploring each application. While participants played with the three widgets, the smartphone's wallpaper changed to a picture of Justin Bieber. Then, the study coordinator distributed four review sheets and gave participants ten minutes to write reviews of the three applications and one of the widgets.

After the review period, participants answered several questions about application behaviors during the review period using the laptop survey. The questions were designed to examine whether participants noticed the misbehaviors and attributed them to any applications, without explicitly priming them to those misbehaviors.

After participants finished these questions, the prompted misbehavior phase of the study began. The order of the misbehaviors matched the previous phase. During the vibration misbehavior, the study coordinator announced:

> Using this computer, we just made one of the three timer apps start vibrating your smartphones. Your next task is to identify which app is responsible. You will have five minutes. Enter your answer in the survey, and continue until you see a page that says, "Stop."

After three minutes, the study coordinator announced:

> If you're unsure which app is responsible, please make a guess and continue.

After completing the survey questions, participants experienced the second misbehavior, and were prompted by the coordinator. When the wallpaper was changed, the study coordinator announced:

> One of the three drawing apps has changed your smartphone's wallpaper. Your next task is to identify which app is responsible. You will have five minutes. Enter your answer in the survey, and continue until you see a page that says, "Stop!"

Three minutes later, the study coordinator announced:

> If you're unsure which app is responsible, please make a guess and continue.

After the full five minutes had passed, the study coordinator announced that the participants were to continue answering the survey questions. At this point, the survey asked direct questions about the attribution mechanisms and the misbehaviors. Upon completion of the survey, we debriefed participants, answered any questions, and then compensated them.

4.5 Participants

In February 2013, we advertised our study on Craigslist in the San Francisco Bay Area. The advertisement stated that we were studying how people interact with applications on Android smartphones. We used an online screening survey to verify potential participants' availability, that they were at least 18 years of age, and to collect data on their personal smartphone's Android version. We offered participants a $35 Visa gift card in exchange for participating in a one hour session. We allowed eligible participants to submit their availability and preferences to attend one of nine sessions.

	Experimental	**Control**
Vibration Misbehavior (without prompting)		
	$n = 37$	$n = 39$
Noticed	21 (57%)	16 (41%)
$\hookrightarrow$*Correctly Identified*	17 (81% of 21)	2 (13% of 16)
Vibration Misbehavior (with prompting)		
	$n = 36$	$n = 39$
Correctly Identified	29 (81%)	12 (31%)

Table 1: Participants in each condition who successfully identified the source of the vibration. The first two rows depict the number of participants who noticed and (of them) correctly identified the source of the misbehavior without first being prompted, whereas the last row depicts the number who correctly identified it after being prompted.

We scheduled 121 Android users to participate, however, our final sample included 76 people.[5] Overall, our sample was 68% male and included ages 19–59 ($\mu = 34.2$, $\sigma = 10.6$). The demographics within each condition closely matched the overall sample (*Control*: 69% male, age: $\mu = 32.8$, $\sigma = 10$; *Experimental*: 68% male, age: $\mu = 35.8$, $\sigma = 11.1$). Our sample represents a range of occupations including accountant, tutor, insurance adjustor, freelancer, executive assistant, and restaurant staff.

5. ANALYSIS

We sought to test two hypotheses with our laboratory experiment:

- H_1: *Experimental* condition participants will be significantly more likely to identify the application responsible for changing the wallpaper.
- H_2: *Experimental* condition participants will be significantly more likely to identify the application responsible for the vibrating.

We asked participants to identify the sources of two misbehaviors: vibrating and changing the wallpaper of the smartphone. The experiment was designed to evaluate whether or not the attribution mechanisms helped participants diagnose the cause of these misbehaviors.

Across both misbehaviors, we observed that participants who noticed the misbehaviors in the *Experimental* condition were significantly more likely to identify the causes of the misbehaviors than participants in the *Control* condition, who did not have access to these attribution mechanisms. In this section, we analyze how participants interacted with each attribution mechanism when explicitly told to pinpoint the source of each misbehavior and whether they noticed the mechanism prior to being prompted.

5.1 Passive Notifications

We instrumented the *Experimental* condition smartphones to display an icon in the status area when we triggered

[5] We anticipated a no-show rate of approximately 20% from prior experience. The seemingly high no-show rate may have been influenced by our policy of denying latecomers.

the vibration misbehavior (Figure 1a). When the notification drawer was expanded by a participant's dragging motion, the notification identified the name of the application blamed for the vibration (Figure 1b).

5.1.1 Attribution

Overall, we found that the vibration notifications were extremely effective when we prompted participants to identify the source of the vibration (Table 1): 29 (81% of 36)[6] participants in the *Experimental* condition were able to correctly identify the application, compared to only 12 (31% of 39) participants in the *Control* condition ($p < 0.0005$, $\phi = 0.500$, one-tailed Fisher's exact test). *Control* participants did no better than random guessing at identifying the responsible application ($\chi^2 = 0.115$, $p = 0.73$, chi-square goodness of fit), whereas *Experimental* participants did significantly better ($\chi^2 = 29.538$, $p < 0.0001$, chi-square goodness of fit). Thus, H_1 is accepted.

Because we asked participants to select the misbehaving application from a multiple choice list of all eight applications that they had reviewed, we expected some participants to answer this question correctly due to random guessing. Of the participants who selected correctly, only 2 (7% of 29) participants in the *Experimental* condition claimed it was due to random guessing, whereas 24 (83% of 29) specifically mentioned interacting with the notifications. Our logs confirmed that 28 correct *Experimental* condition participants (97% of 29) expanded the notification drawer immediately after the misbehavior occurred. In the *Control* condition, 5 participants (42% of 12), a plurality, specifically mentioned arriving at the correct answer by random guessing. The explanations from the other correct *Control* participants suggest that all but two correct responses may also be explained by chance:

- *"Nothing else has a reason to vibrate"*
- *"Notification at the top"*
- *"There was a notification at the top of the screen that said time up"*
- *"Because it was set to vibrate at 00:00:00"*

A total of five *Control* condition participants attributed their selections to the notification drawer (three were correct, two were incorrect), though these participants did not see the attribution notifications that we created for the *Experimental* condition participants. Instead, these participants viewed the notifications generated by the individual timer applications, and by chance, in two of these cases, these happened to be the correct applications to which we attributed the vibration. Our logs indicate that after being prompted about the vibration misbehavior, twenty-seven *Control* condition participants (69% of 39) expanded the notification drawer in hopes of identifying the misbehaving application. However, without system-generated notifications to attribute behaviors to particular applications, the only way of identifying the cause of an ongoing misbehavior is by individually killing applications until the misbehavior stops. Using this method, only two of the *Control* condition participants correctly identified the sources of the misbehavior.

During this task, we asked participants to evaluate applications that all had the ability to vibrate the smartphone in order to make the source of the vibration ambiguous. If a participant based her decision on an application's granted permissions, as shown in the Settings application (Figure 3), she would have found that all three of the timer applications had been granted permission to vibrate. Nonetheless, six participants in the *Control* condition identified applications by incorrectly stating that their selected applications were the only applications with the ability to vibrate the smartphone. From our log files, we can confirm that none of these participants ever viewed the panel in the Settings application that listed this information, and therefore these participants were inferring each application's granted permissions based on its visible features (as determined by interacting with the application during the earlier phases of the experiment). Specifically, these participants erroneously assumed that an application's abilities could be determined based on the options available in the user interface. This result corroborates Felt et al.'s finding that very few users view permissions and instead determine an application's abilities based on either their familiarity with it or reviews [17].

Of the 34 participants between both conditions who could not correctly identify the misbehaving application, the most common strategy was random guessing (38% of 34), while the second most common strategy was to choose the application that was in use at the time that the misbehavior occurred (29% of 34). This finding corroborates our online survey results (Section 3): many users are unaware that applications running in the background have the same abilities as applications running in the foreground.

Finally, we asked participants to report the confidence of their selections using a five-point Likert scale ("very confident" to "very unconfident," anchored around "unsure"). Those in the *Experimental* condition reported a median of five (i.e., "very confident"), whereas those in the *Control* condition reported a median of three ("unsure"); this difference was statistically significant ($U = 330.0$, $p < 0.0005$). More importantly, only in the *Experimental* condition were correct answers correlated with a high degree of confidence ($\rho = 0.526$, $p < 0.0005$). We observed no such correlation in the *Control* condition.

While it is clear that participants understood how to use the attribution notification when we prompted them to pinpoint the source of the misbehavior, our next question was whether or not they sought out this information, and reacted to it, on their own initiative.

5.1.2 Exploration of the Notification

During the first phase of the experiment, when we did not prompt participants, we caused the phones to vibrate while the participants were exploring the fourth application (i.e., while all four applications were running) so as to introduce ambiguity as to which application was responsible. As a proxy for whether or not participants noticed the alleged misbehavior, we examined whether or not they mentioned anything about the phone vibrating either in their application reviews or in their survey responses.[7] Overall,

[6] Due to technical problems, one *Experimental* condition participant did not receive the vibration misbehavior when we prompted them.

[7] After participants turned in their review sheets, they answered survey questions about whether any of the applications behaved unexpectedly, such as "crash or stop functioning," "unexpectedly access phone hardware (e.g., flashlight, camera, vibrate, microphone, etc.)," or "unexpectedly change system settings (e.g., wallpaper, desktop icons, favorites, etc.)." Our goal was to determine whether partici-

	Experimental	**Control**
Wallpaper Misbehavior (without prompting)		
	$n = 35$	$n = 39$
Noticed	5 (14%)	8 (21%)
↪*Correctly Identified*	0 (0% of 5)	0 (0% of 8)
Wallpaper Misbehavior (with prompting)		
	$n = 35$	$n = 38$
Correctly Identified	12 (34%)	3 (8%)

Table 2: Participants in each condition who successfully identified the source of the wallpaper change. The first two rows depict the number of participants who noticed and (of them) correctly identified the source of the misbehavior without first being prompted, whereas the last row depicts the number who correctly identified it after being prompted.

thirty-seven participants (49% of 76) noticed the vibration misbehavior.

Of the thirty-seven participants who noticed the vibration, twenty-two attributed it to a specific application in their reviews, twelve named a specific application in the survey (and not the reviews), and three noted the vibration, but did not identify the application that they believed to be responsible (e.g., they indicated the misbehavior on multiple reviews). We observed that participants in the *Experimental* condition were significantly more likely to use the review forms to attribute the misbehavior to a particular application ($p < 0.0205$; one-tailed Fisher's exact test). We believe this indicates that the participants who viewed the notifications were more confident in their identifications, and therefore more willing to "name names" in the reviews, so to speak. Moreover, these participants were more likely to be correct: without being prompted, seventeen *Experimental* condition participants (81% of 21) and two *Control* condition participants (13% of 16) correctly identified the application that caused the misbehavior ($p < 0.0005$, Fisher's exact test, $\phi = 0.678$). Our logs confirm that all 17 *Experimental* condition participants expanded the notification drawer immediately after the vibration misbehavior.

5.2 Settings with Provenance

We used notifications to represent the vibration misbehavior because we believed that they were well-suited to convey information about an ongoing annoyance. However, we also wanted to examine whether annotations in Settings would be successful at informing users about one-time misbehaviors with a lasting effect, such as changes to system settings. We hypothesized that adding these attributions in Settings would be intuitive to users because this is where they would normally go to undo an undesirable change. In our experiment, when a randomly-selected application changed the smartphone's wallpaper, *Experimental* condition participants could find the name of the application that caused the change by viewing either the Display Settings or the Wallpaper Chooser (Figure 2).

5.2.1 Attribution

When we explicitly prompted participants to identify the application that allegedly changed the smartphone's wallpaper, we found that the Display Settings and Wallpaper Chooser annotations were significantly more effective than no annotations (Table 2): twelve participants (34% of 35)[8] in the *Experimental* condition named the correct application compared to only three participants (8% of 38) in the *Control* condition ($p < 0.006$, $\phi = 0.326$, one-tailed Fisher's exact test). Thus, H_2 is accepted.

Of the twelve *Experimental* condition participants who chose the correct application after we explicitly prompted them, ten (83% of 12) said that they based their selections on the attribution mechanisms. The logs confirmed that all ten viewed one of the two attribution mechanisms after we explicitly prompted them to identify the source of the misbehavior: three viewed the annotated Wallpaper Chooser, six viewed the Display Settings, and one participant viewed both attribution mechanisms. As before, we expected a certain number of "false positives"—correct answers due to guessing—in each condition. Unlike the previous task, in which *Control* condition participants could still identify the source of the ongoing vibration misbehavior by terminating applications until the misbehavior stopped, there was no way of identifying the cause of a one-time system change, like changing the wallpaper, on these Android devices. Two *Experimental* condition participants—who did not notice the annotations in Settings—and three *Control* condition participants selected the correct application by chance:

- *"I'm guessing, I'm not sure"*
- *"None were running when the wallpaper changed, but Coloring Princess seems very poorly coded, contained ads, and had an option to set wallpapers. This would be my bet."*
- *"Based it on the icon of the app, [which] had a celeb on it, so the wall paper also changed to a celeb."*
- *"I guessed because it happened after I used the app previously"*
- *"It asks do you want to save or set as wallpaper"*

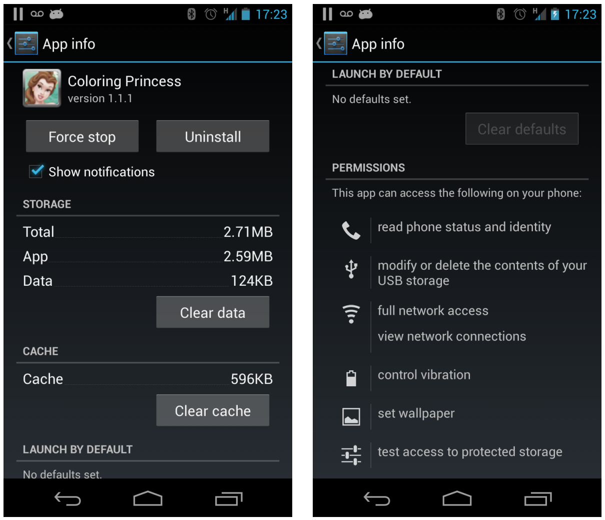
Of the fifty-eight participants across both conditions who could not identify the source of the wallpaper change, the most common strategies were random guessing (33% of 58) and blaming the application that was in the foreground at the time that the misbehavior occurred (33% of 58). Across the two conditions, participants' selection strategies did not significantly differ. Similar to the previous task, several participants—six of the fifty-eight—indicated that they made their selections based on the application's visible features related to the wallpaper:

- *"Coloring Princess is the only app which allows one to set a picture or drawing as wallpaper."*
- *"I am not sure between Sketch Free 2 and Coloring Princess: both provided an option in the settings menu to set the image as wallpaper."*
- *"I can save it into the desktop as wallpaper."*
- *"I thought I remembered seeing a wallpaper option."*

As can be seen from these quotes, many participants inferred each application's ability to change the wallpaper based on whether or not such a feature appeared in the application's user interface. In truth, the only way of determining an application's abilities is via the Apps sub-panel

pants had noticed the misbehaviors without priming them.

[8] Due to technical difficulties, two participants in the *Experimental* condition received neither of the wallpaper misbehaviors, and one participant in the *Control* condition did not receive the wallpaper misbehavior after we prompted them.



17:23

App info

Coloring Princess
version 1.1.1

Force stop | Uninstall

☑ Show notifications

STORAGE

Total	2.71MB
App	2.59MB
Data	124KB

Clear data

CACHE

Cache	596KB

Clear cache

LAUNCH BY DEFAULT

No defaults set.

(a) *First Page*

17:23

App info

LAUNCH BY DEFAULT

No defaults set.

Clear defaults

PERMISSIONS

This app can access the following on your phone:

- read phone status and identity
- modify or delete the contents of your USB storage
- full network access
- view network connections
- control vibration
- set wallpaper
- test access to protected storage

(b) *Second Page*

Figure 3: The application information in the Settings application (a). Users can only see permission information if they know to scroll below the fold (b).

within the Settings application, which lists all of the permissions declared in the application's manifest file. From our log files, only eight participants (14% of 58) could have viewed this information. However, this is likely an upper bound on the number who viewed permissions for the following reasons:

1. The Android device logs only show us that participants accessed the Apps sub-panel of the Settings application, and not whether they further selected a specific application in order to view its permissions (logging this would have required us to replace Settings with our own application, rather than using the existing debugging facility).
2. If they viewed a specific application's abilities, we could not capture its name, and therefore we do not know whether they looked at the permissions for one application or many.
3. When an application is selected from Settings, its permissions are not shown unless the user knows to scroll to the bottom of the page; by default, other information about the application fills the screen (Figure 3).

Furthermore, if participants did determine whether an application had the ability to change the wallpaper based on the permissions shown in the Settings application, this would have narrowed the list of possible culprits down from the eight multiple-choice options to the three applications that had this ability. That is, without the attribution mechanism provided to *Experimental* condition participants, there was no way of identifying the misbehaving application with certainty. At the same time, despite the significant improvement in the *Experimental* condition, only a minority of participants were able to correctly identify the application responsible for the misbehavior. Upon examining our logs, we concluded that the majority of participants did not encounter either attribution mechanism: a total of fourteen participants encountered either the annotated Display Settings or the annotated Wallpaper Chooser, ten of whom (95%CI: [42%, 92%]) based their decisions on the information in these attribution mechanisms.

Finally, as we observed with the vibration misbehavior, correctly identifying the application responsible for the misbehavior was highly correlated with participants' confidence in those responses among participants in the *Experimental* condition ($\rho = 0.663$, $p < 0.0005$). We did not observe this correlation among participants in the *Control* condition.

5.2.2 *Exploration of the Settings Annotations*

To gain insight into why the attribution rates were so low during the wallpaper misbehavior, we examined whether participants noticed and reacted to the misbehavior prior to our prompts. Prior to explicitly prompting participants to identify the cause of the wallpaper change misbehavior, we discretely triggered the misbehavior during the application evaluation task, and observed whether they mentioned it in either their reviews or the computer-based survey. Overall, a total of 13 participants (18% of 74; 5 in the *Experimental* condition and 8 in the *Control* condition) noticed the misbehavior. However, not a single participant was able to correctly attribute the misbehavior to a specific application.

Examining the smartphone logs, we found that without prompting about the misbehavior, not a single participant in the *Experimental* condition encountered either of the two attribution mechanisms that would have allowed them to identify the source of the misbehavior. Instead, ten of the thirteen participants (77%) who noticed the misbehavior simply attributed it to the application that they were using at the time. Thus, while these attribution mechanisms were effective for users who encountered them, they were unhelpful when participants failed to notice them.

6. DISCUSSION

Through our experiments, we observed that many participants assumed that unexpected events on their smartphones can be attributed to the application running in the foreground; they did not realize that applications running in the background may have the same abilities. Furthermore, participants made assumptions about a given application's abilities—and therefore its potential for misbehavior—based on the features that were visible to them, rather than the permissions that it had already been granted. At the same time, we observed that participants who noticed misbehaviors and who had access to attribution mechanisms were significantly more likely to correctly identify the sources of misbehaviors. In this section, we discuss some possible explanations for participants' behaviors, limitations of our experiments, and conclude with future work.

6.1 Explanations

We discuss participants' potential biases, learning curves, and how our results are likely applicable to other platforms.

6.1.1 *Application Familiarity*

We intentionally asked participants to evaluate applications that we hoped would be unfamiliar to them. If participants in one condition were more familiar with an application than participants in the other condition, they might be biased towards or against attributing misbehavior to the familiar application. To test for this bias in the exit survey, we asked participants to declare their familiarity with each application using a five-point Likert scale. We performed a Mann-Whitney U test between the two conditions, comparing the median familiarity scores, and correcting for multiple testing. We observed no significant differences between conditions with regard to familiarity with any of the applications. Thus, if there was a bias toward or against any of the applications, it did not observably differ between the two conditions.

6.1.2 *Time Heals All Wounds*

While participants performed significantly better when we provided them with attribution mechanisms, we were surprised at their relatively low rate of success at identifying the source of the wallpaper misbehavior. In the *Experimental* condition, only 32% of participants (95%CI: [18%, 50%]) were able to correctly identify the application that changed the wallpaper (once they were made aware of the misbehavior). In contrast, 78% of these participants (95%CI: [62%, 90%]) were able to successfully use the notification drawer to correctly identify the cause of the vibration misbehavior. This contrast was statistically significant ($Z = -3.900$, $p < 0.0005$, Wilcoxon Signed Ranks test).

One possible explanation for this is that participants were simply more familiar with seeing notifications in the status area and therefore were more likely to look to this area for potential clues when trying to determine the cause of each misbehavior. Our online survey, qualitative experiment, and experimental log data suggest that the vast majority of Android users understand that the status bar provides information about application activity and that it is interactive (e.g., 94.7% of our laboratory participants pulled down the notification drawer at least once during the experiment). In contrast to notifications, the annotations that we introduced to identify the cause of the wallpaper misbehavior have not been used before, and therefore it is unclear whether participants knew to look for these.

We believe that this modest success rate may be of little concern for two reasons. First, if platform designers were to introduce similar annotations, users will likely become more aware of them over time, just as they did after Android introduced a Data Usage panel in the Settings application. Second, not every user needs to be aware of attribution mechanisms for them to have an impact. Felt et al. showed that most users base their decisions on application reviews [17]. Thus, an application's misbehavior will likely become widely disseminated after a small number of savvy users note it in their reviews. This may be sufficient to protect most users from misbehaving applications and to incentivize developers to avoid such misbehaviors.

6.1.3 *Other Platforms*

We specifically evaluated attribution mechanisms under the Android platform using existing Android users. However, we believe that many of our findings are also relevant to other platforms. For instance, users of iOS 6 can view the applications that may have accessed their location, contacts, calendars, reminders, photos, and Bluetooth [8]. Our results suggest that this addition has benefited users.

More broadly, our results support Felt et al.'s permission model [14]: eliminating requests for permissions that only represent low-level risks (e.g., annoyances) or whose effects can be fully reversed, so long as users are provided with attribution mechanisms that help them identify the source of a misbehavior. The attribution mechanisms that we evaluated in this study appear to be successful at doing just that.

6.2 Limitations

As with all controlled experiments, certain factors may have skewed our results.

6.2.1 *Moral Hazard*

One possible explanation for participants' behavior is moral hazard: because they were not using their own mobile devices during the experiment, it is possible that they were not sufficiently motivated to identify the causes of the misbehaviors. For instance, this may explain why so few participants acknowledged the misbehaviors prior to our prompting. The only way of controlling for this factor would be by instrumenting participants' own smartphones that they regularly use and observing them over a longer period of time.

6.2.2 *Multiple Notifications*

Another problem we observed was that during the evaluation phase, some of the timer applications used notifications to display information (e.g., amount of time left). Several participants in the *Experimental* condition noticed these application-drawn notifications alongside our system-drawn notification. It is possible that this created confusion. If system-drawn notifications are to be successfully used to attribute application behaviors, they should be styled in such a way that they are readily distinguishable from application-drawn notifications [32].

6.2.3 *Imprecise Logging*

In cases where modifying Android components proved exceptionally difficult, we used Android's existing logging information. This allowed us to see *whether* a user opened a particular settings panel, but it did not show us what they did within that settings panel (e.g., they may have viewed one or more sub-panels). This was useful in corroborating participants' reported behaviors, but it prevented us from relying on the logs alone to explain responses. Likewise, while we can definitively say whether they pulled down the notification drawer or viewed a settings panel, without performing eye tracking, it is impossible for us to say whether or not they actually read the corresponding text.

6.2.4 *Technical Difficulties*

Finally, as we noted in our analysis earlier, due to technical problems, not every participant received all four misbehaviors (i.e., two without prompting followed by two with prompting). In our analysis, we noted two specific participant behaviors that resulted in problems: accidentally killing the study monitor process and rebooting the smartphone.

While attempting to identify the cause of the vibration, several participants arbitrarily terminated running applications. In six cases, this included our study monitor application, which initiated the misbehaviors and attributed them to other applications. This meant that these participants' smartphones did not misbehave correctly. This was more likely to be a problem during the wallpaper misbehavior, because participants generally killed applications only during the vibration misbehavior (likely since it was ongoing rather than a one-time change). In one of these cases, a participant showed us that her phone was not vibrating after we prompted her about the misbehavior, which resulted in us being able to restart it and still collect data for this misbehavior (though she did not receive the earlier wallpaper misbehavior when we did not prompt participants). In four other cases when the prompted vibration misbehavior did not occur, our logs confirmed that all of these participants were in the *Experimental* condition. All of these participants pulled down the notification drawer and blamed the same application that had caused the vibration misbehavior previously before prompting. Thus, we marked these responses as being correct.

Learning from this oversight, we should have made our study monitor application a system service, and designed it to be more robust to these edge cases.

6.3 Future Work

Our results demonstrate that attribution mechanisms can be a powerful tool to assist users in identifying misbehaving applications. However, future work is needed to show how to present such mechanisms more effectively.

The mechanisms we examined in this study were significantly more effective at attributing the source of vibration misbehaviors and wallpaper changes for users who noticed the misbehaviors, as compared to the status quo (i.e., no attribution mechanisms). That said, every conceivable misbehavior may not always have an attribution mechanism, as this would rapidly become overwhelming to the user. More importantly, too many notifications that users find unimportant will rapidly habituate them to ignoring all notifications. Likewise, too many options and text within the Settings application may overwhelm the users. Thus, we are faced with the problem of separating the unimportant misbehaviors, which do not necessarily need to be attributed, from the important ones that do. Building on Felt et al.'s work [15], we asked laboratory participants under which circumstances would they would like to see notifications or annotations to settings panels in the future.

We asked participants which of the following 17 permissions they would like to see annotations in Settings for:

1. Battery usage (71.1%)
2. Data usage (68.4%)
3. Location access (65.8%)
4. Connected/disconnected WiFi (55.3%)
5. Placed calls (55.3%)
6. Sent SMS (53.9%)
7. Storage usage (50.0%)
8. Photo library access (48.7%)
9. Date/time changed (47.4%)
10. Contact list (address book) access (43.4%)
11. Volume changed (40.8%)
12. Wallpaper changed (38.2%)
13. Read device ID (36.8%)
14. Calendar access (35.5%)
15. Recently paired via Bluetooth (28.9%)
16. Ringtone changed (27.6%)
17. Font changed (26.3%)

The top two items for which participants wanted to see annotations in Settings were battery and data usage. All participants in our laboratory experiment were Android users and therefore already had a battery usage attribution mechanism within the Settings application, and 30 of them (39% of 76) used version 4.0 or later, and therefore also had a data usage attribution mechanism. Another interesting finding from this data is that none of the new attribution mechanisms included in iOS 6 [8] appear at the top of the list. Thus, while Apple should be applauded for attempting to increase transparency by adding new attribution mechanisms, it is not clear whether or not the chosen permissions are of greatest concern to users.

With regard to notifications, we asked participants to choose from a list of 18 permissions:

1. Vibration (57.9%)
2. Connect/Disconnect WiFi (56.6%)
3. Quit other apps (46.1%)
4. Set alarm clock (42.1%)
5. Connect/Disconnect Bluetooth (40.8%)
6. Playing audio (32.9%)
7. Send call to voicemail (32.9%)
8. Recording audio (28.9%)
9. Disconnect call (28.9%)
10. Recording video (27.6%)
11. Flashlight (27.6%)
12. Change wallpaper (25.0%)
13. Read device ID (22.4%)
14. Change time (19.7%)
15. Change font (14.5%)
16. Change ringtone (14.5%)
17. Modify dictionary (10.5%)

The ability that participants most wanted to see in future notifications was the ability to vibrate the device—the very notification that *Experimental* participants saw during the experiment (58% of 76). We question this particular result, since participants may have been biased by the experimental design to have a stronger reaction to this misbehavior, as they had just experienced it (it is possible that the desire for an annotation in Settings for recent wallpaper changes was also biased). Thus, future work is needed to determine which permissions should be implicitly granted and represented with attribution mechanisms.

7. ACKNOWLEDGMENTS

We would like to thank Adrienne Porter Felt, Rowilma del Castillo, Miho Tanaka, and Refjohürs Lykkewe. This work was supported by the Intel Science and Technology Center for Secure Computing and the Air Force Office of Scientific Research (Grant #29182280-51677-C).

8. REFERENCES

[1] Android Open Source Project (AOSP). `http://source.android.com/`.

[2] iOS App Programming Guide: App States and Multitasking. `https://developer.apple.com/library/IOs/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html`.

[3] T. S. Amer and J. B. Maris. Signal words and signal icons in application control and information technology exception messages – hazard matching and habituation effects. Technical Report Working Paper

Series–06-05, Northern Arizona University, Flagstaff, AZ, October 2006. http://www.cba.nau.edu/Faculty/Intellectual/workingpapers/pdf/Amer_JIS.pdf.

[4] R. Böhme and J. Grossklags. The security cost of cheap user interaction. In *Proceedings of the 2011 New Security Paradigms Workshop*, NSPW '11, pages 67–82, New York, NY, USA, 2011. ACM.

[5] J. C. Brustoloni and R. Villamarín-Salomón. Improving security decisions with polymorphic and audited dialogs. In *Proceedings of the 3rd Symposium on Usable Privacy and Security*, SOUPS '07, pages 76–85, New York, NY, USA, 2007. ACM Press.

[6] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani. Crowdroid: behavior-based malware detection system for Android. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '11, pages 15–26, New York, NY, USA, 2011. ACM.

[7] J. Cheng, S. H. Wong, H. Yang, and S. Lu. SmartSiren: virus detection and alert for smartphones. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, MobiSys '07, pages 258–271, New York, NY, USA, 2007. ACM.

[8] J. Cipriani. How to control your privacy settings in iOS 6. http://howto.cnet.com/8301-11310_39-57507698-285/how-to-control-your-privacy-settings-on-ios-6/, September 19 2012. Accessed: March 6, 2013.

[9] L. Davi, A. Dmitrienko, A.-R. Sadeghi, and M. Winandy. Privilege escalation attacks on Android. In *Proceedings of the 13th International Conference on Information Security*, ISC'10, pages 346–360, Berlin, Heidelberg, 2011. Springer-Verlag.

[10] S. Egelman, L. F. Cranor, and J. Hong. You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In *Proceeding of The 26th SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1065–1074, New York, NY, USA, 2008. ACM.

[11] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI '10, pages 1–6, Berkeley, CA, USA, 2010. USENIX Association.

[12] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A study of Android application security. In *Proceedings of the 20th USENIX Security Symposium*, SEC '11, pages 21–21, Berkeley, CA, USA, 2011. USENIX Association.

[13] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 627–638, New York, NY, USA, 2011. ACM.

[14] A. P. Felt, S. Egelman, M. Finifter, D. Akhawe, and D. Wagner. How to ask for permission. In *Proceedings of the 7th USENIX Workshop on Hot Topics in Security*, HotSec '12, pages 7–7, Berkeley, CA, USA, 2012. USENIX Association.

[15] A. P. Felt, S. Egelman, and D. Wagner. I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In *Proceedings of the 2nd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '12, pages 33–44, New York, NY, USA, 2012. ACM.

[16] A. P. Felt, K. Greenwood, and D. Wagner. The effectiveness of application permissions. In *Proceedings of the 2nd USENIX Conference on Web Application Development*, WebApps '11, pages 7–7, Berkeley, CA, USA, 2011. USENIX Association.

[17] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android permissions: user attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 3:1–3:14, New York, NY, USA, 2012. ACM.

[18] A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin. Permission re-delegation: attacks and defenses. In *Proceedings of the 20th USENIX Security Symposium*, SEC '11, pages 22–22, Berkeley, CA, USA, 2011. USENIX Association.

[19] S. Flosi. comScore Reports October 2012 U.S. Mobile Subscriber Market Share, November 30 2012. http://www.comscore.com/Insights/Press_Releases/2012/11/comScore_Reports_October_2012_U.S._Mobile_Subscriber_Market_Share.

[20] C. Guo, H. J. Wang, and W. Zhu. Smart phone attacks and defenses. In *ACM Workshop on Hot Topics in Networks*, HotNets '04, 2004.

[21] J. Howell and S. Schechter. What you see is what they get: Protecting users from unwanted use of microphones, cameras, and other sensors. In *Proceedings of the 2010 Workshop on Web 2.0 Security and Privacy (W2SP)*, 2010. http://w2spconf.com/2010/papers/p05.pdf.

[22] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall. A conundrum of permissions: installing applications on an Android smartphone. In *Proceedings of the 16th International Conference on Financial Cryptography and Data Security*, FC '12, pages 68–79, Berlin, Heidelberg, 2012. Springer-Verlag.

[23] S. Kim and M. Wogalter. Habituation, dishabituation, and recovery effects in visual warnings. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 53, pages 1612–1616. SAGE Publications, 2009.

[24] K. R. Laughery and A. Hammond. Overview. In M. S. Wogalter, D. M. DeJoy, and K. R. Laughery, editors, *Warnings and Risk Communication*, chapter 1, pages 2–11. Taylor and Francis, London, 1999.

[25] C. Lever, M. Antonakakis, B. Reaves, P. Traynor, and W. Lee. The Core of the Matter: Analyzing malicious traffic in cellular carriers. In *Proceedings of the ISOC Network & Distributed Systems Security Symposium*, NDSS '13, 2013.

[26] G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos. Paranoid Android: versatile protection for smartphones. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 347–356, New York, NY, USA, 2010. ACM.

[27] F. Roesner, T. Kohno, A. Moshchuk, B. Parno, H. J. Wang, and C. Cowan. User-driven access control: Rethinking permission granting in modern operating systems. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 224–238, Washington, DC, USA, 2012. IEEE Computer Society.

[28] M. Stiegler and M. S. Miller. A capability-base client: The DarpaBrowser. Technical report, Combex Inc., June 2002. `http://www.combex.com/papers/darpa-report/index.html`.

[29] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor. Crying wolf: an empirical study of SSL warning effectiveness. In *Proceedings of the 18th USENIX Security Symposium*, SEC '09, pages 399–416, Berkeley, CA, USA, 2009. USENIX Association.

[30] T. Vidas, D. Votipka, and N. Christin. All your droid are belong to us: a survey of current Android attacks. In *Proceedings of the 5th USENIX conference on Offensive technologies*, WOOT '11, pages 10–10, Berkeley, CA, USA, 2011. USENIX Association.

[31] H. Xia and J. C. Brustoloni. Hardening web browsers against man-in-the-middle and eavesdropping attacks. In *Proceedings of the 14th International Conference on the World Wide Web*, WWW '05, pages 489–498, New York, NY, USA, 2005. ACM.

[32] K.-P. Yee. User interaction design for secure systems. In *Proceedings of the 4th International Conference on Information and Communications Security*, ICICS '02, pages 278–290, London, UK, UK, 2002. Springer-Verlag.

APPENDIX

A. LAB EXPERIMENT SURVEY

A.1 Implicit Misbehaviors

1. Did any of the apps crash or stop functioning while you were using them?
   - Yes
   - No
   - I'm not sure
2. Please select the app(s) that crashed or stopped functioning:
   - Ovo Timer
   - Easy Stop Watch & Timer
   - Dice
   - PicsArt Kaleidoscope
   - Sketch Free 2
   - Coloring Princess
   - One or more of the widgets
3. How do you know that this/these app(s) crashed or stopped functioning?
4. Did any of the apps unexpectedly change system settings (e.g., wallpaper, desktop icons, favorites, etc.)?
   - Yes
   - No
   - I'm not sure
5. Please select the app(s) that unexpectedly changed system settings (e.g., wallpaper, desktop icons, favorites, etc.):
   - Ovo Timer
   - Easy Stop Watch & Timer
   - Dice
   - PicsArt Kaleidoscope
   - Sketch Free 2
   - Coloring Princess
   - One or more of the widgets
6. How do you know that this/these app(s) were responsible for unexpectedly changing system settings (e.g., wallpaper, desktop icons, favorites, etc.)?
7. Which system settings unexpectedly changed?
8. Did any of the apps unexpectedly access phone hardware (e.g., flashlight, camera, vibrate, microphone, etc.)?
   - Yes
   - No
   - I'm not sure
9. Please select the app(s) that unexpectedly accessed phone hardware (e.g., flashlight, camera, vibrate, microphone, etc.):
   - Ovo Timer
   - Easy Stop Watch & Timer
   - Dice
   - PicsArt Kaleidoscope
   - Sketch Free 2
   - Coloring Princess
   - One or more of the widgets
10. How do you know that this/these app(s) were responsible for unexpectedly accessing phone hardware (e.g., flashlight, camera, vibrate, microphone, etc.)?
11. What phone hardware was unexpectedly used?

A.2 Explicit Vibration

1. Which app do you believe is responsible for vibrating the phone just now?
   - Ovo Timer
   - Easy Stop Watch & Timer
   - Dice
   - PicsArt Kaleidoscope
   - Sketch Free 2
   - Coloring Princess
   - One or more of the widgets
2. How did you determine which app made the phone vibrate?
3. How confident are you that this was the responsible app?
   - Very confident
   - Confident
   - Unsure
   - Unconfident
   - Very unconfident

A.3 Explicit Wallpaper Changed

1. Which app do you believe is responsible for changing the phone's wallpaper just now?
   - Ovo Timer
   - Easy Stop Watch & Timer
   - Dice
   - PicsArt Kaleidoscope
   - Sketch Free 2
   - Coloring Princess
   - One or more of the widgets
2. How did you determine which app changed the wallpaper?
3. How confident are you that this was the responsible app?
   - Very confident
   - Confident
   - Unsure
   - Unconfident
   - Very unconfident

A.4 Exit Survey

A.4.1 Notification Questions

1. Did you see this icon/notification during the experiment (see Figure 1)?
   - Yes
   - No
   - I'm not sure / I don't remember
2. What do you believe this icon/notification means?
3. Similar notifications can be used to tell you how apps are using your phone. Which of the following actions would you like to see notifications for?
   - Vibration
   - Change wallpaper
   - Quit other apps
   - Playing audio
   - Change time
   - Connect/disconnect Bluetooth
   - Bluetooth
   - Recording audio
   - Recording video
   - Taking pictures
   - Flashlight
   - Set alarm clock
   - Modify dictionary
   - Change font
   - Change ringtone
   - Connect/disconnect
   - WiFi
   - Read device ID
   - Send call to voicemail
   - Disconnect call

A.4.2 Settings Audit Questions

1. Did you see this information during the experiment (see Figure 2)?
   - Yes
   - No
   - I'm not sure / I don't remember
2. The Settings app is sometimes used to show the user information about the phone, such as which app most recently modified a setting or used a resource. Would you like to see information in Settings about which apps did any of the following:
   - Data usage
   - Battery usage
   - Recently paired via Bluetooth
   - Connected/Disconnected WiFi
   - Changed volume
   - Changed wallpaper
   - Storage usage
   - Accessed location (GPS)
   - Accessed contact list (address book)
   - Changed date/time
   - Changed font
   - Changed ringtone
   - Accessed calendar(s)
   - Accessed photos
   - Read device ID
   - Placed calls
   - Sent SMS

A.4.3 Demographic Information

1. For each of the following apps that you used in this experiment, please indicate how familiar you were with each app prior to this experiment. Options listed in a table: use daily, have used a few times, have used once, have never used, but have heard of it, and have never heard of it.
   - Ovo Timer
   - Easy Stop Watch & Timer
   - Dice
   - PicsArt Kaleidoscope
   - Sketch Free 2
   - Coloring Princess
   - One or more of the widgets
2. What is the make/model of your Android device?
3. How long have you owned your current Android device?
   - Less than six months
   - 6–12 months
   - 12–24 months
   - Over 24 months
4. What version of Android is installed on your smartphone? If you do not know, the experimenter can help you determine this.
5. In what year were you born?
6. What is your gender?
7. What is your occupation?

A.5 Debriefing

In this study, you were asked to write reviews about several Android applications. During that process, the provided smartphone may have started to misbehave by vibrating and/or changing the desktop wallpaper. The true purpose of the experiment was to see what actions you would take to determine the cause of this behavior. Some of you were provided with icons and settings panels that helped attribute these behaviors to particular applications that you had recently used; our goal was to assess whether or not you noticed these indicators and whether or not they were helpful to you. In reality, none of the applications you used were responsible for these behaviors, and the reviewing task was subterfuge.

The data that you contributed to this project will help us to improve the usability of smartphone notifications for all users.