The Effect of Developer-Specified Explanations for Permission Requests on Smartphone User Behavior

Joshua Tan[1], **Khanh Nguyen**[2], **Michael Theodorides**[3], **Heidi Negrón-Arroyo**[4], **Christopher Thompson**[3], **Serge Egelman**[3], **David Wagner**[3]

[1]North Dakota State University, joshua.tan@ndsu.edu
[2]University of California, Riverside, knguy068@ucr.edu
[3]University of California, Berkeley, {theodorides,cthompson,egelman,daw}@cs.berkeley.edu
[4]University of Puerto Rico, Mayagüez, heidi.negron1@upr.edu

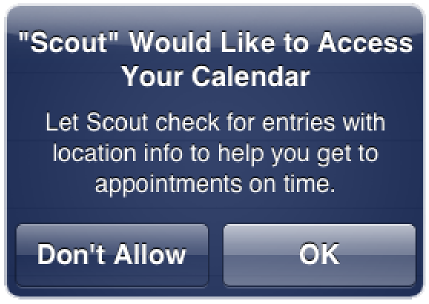
ABSTRACT
In Apple's iOS 6, when an app requires access to a protected resource (e.g., location or photos), the user is prompted with a permission request that she can allow or deny. These permission request dialogs include space for developers to optionally include strings of text to explain to the user why access to the resource is needed. We examine how app developers are using this mechanism and the effect that it has on user behavior. Through an online survey of 772 smartphone users, we show that permission requests that include explanations are significantly more likely to be approved. At the same time, our analysis of 4,400 iOS apps shows that the adoption rate of this feature by developers is relatively small: around 19% of permission requests include developer-specified explanations. Finally, we surveyed 30 iOS developers to better understand why they do or do not use this feature.

Author Keywords
Smartphones; Privacy; Access Control; Usability

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces; D.4.6 Operating Systems: Security and Protection

INTRODUCTION
In 2012, the number of smartphone users crossed 1 billion, and that number is expected to double by 2015 [26]. Smartphones allow third party apps to access sensor data (e.g., location, photos, etc.) and personal information (e.g., address book contacts, calendar appointments, etc.) in order to create rich user experiences and subsidize costs through targeted advertising. To empower users to make informed choices about how their data is used, each smartphone platform has a method for allowing users to grant permission for an app to access certain types of data.

CHI 2014, April 26–May 1, 2014, Toronto, Ontario, Canada.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2473-1/14/04...$15.00.
http://dx.doi.org/10.1145/2556288.2557400


"Scout" Would Like to Access Your Calendar

Let Scout check for entries with location info to help you get to appointments on time.

Figure 1. A calendar permission request for the Scout app including a developer-specified purpose string explaining the reason for the request.

In Apple's iOS 6 (released in September 2012), permission request dialogs appear at runtime whenever an app first requests access to any of the following six resources: the user's geophysical location, address book contacts, photos and videos, calendars, reminders, and Bluetooth pairing [3]. To make these requests more understandable to users, developers can optionally specify a *purpose string* to explain why an app needs access to a particular resource. Figure 1 shows an example of a calendar request with such a purpose string.

We studied the effects of developer-specified purpose strings through three different experiments. First, we performed an online experiment on 772 smartphone users to measure their willingness to grant access to protected resources. We observed that on average, the rate at which participants approved permission requests increased by 12% when purpose strings were present. Surprisingly, the actual content of the purpose strings did not have an observable effect (i.e., participants were significantly more likely to approve requests that included purpose strings, regardless of what those requests said). Second, we performed an analysis of 4,400 iOS apps to measure how many are taking advantage of this feature and whether there are any trends with regard to developers' text choices. Finally, we performed a qualitative survey of 30 iOS developers to examine their opinions of this feature, including why they choose (or choose not) to use it. Taken together, our experiments show that purpose strings can lead users to grant more requests (even if they do not convey useful information), but few apps employ them, due to developers being unaware of the feature or its impact on user behavior.

BACKGROUND AND RELATED WORK

Research has shown that with regard to mobile device security and privacy, malicious apps are rare [17]. A more widespread threat appears to be apps that, possibly unintentionally, access users' personal data in ways that users neither expect nor condone. Enck *et al.* showed that there is systematic misuse of personal and phone identifiers stored on Android smartphones [10]. Researchers have developed techniques to track the flow of privacy-sensitive data and detect when it leaves the device [9] and to identify overprivileged applications [11]. Because users may weigh the trade-off between privacy and functionality differently, many mobile platforms allow users to decide whether third party apps can access certain protected resources.

When choosing how to represent permission requests, platform developers have essentially four mechanisms at their disposal: automatically granted permissions, trusted UI components, runtime warnings, and install-time warnings [14]. Android uses install-time warnings for all of its permission requests. However, Felt *et al.* found that a majority of Android users do not look at permissions during installation, and very few could explain specific permissions when asked [12].

Instead of presenting a list of all requested permissions at installation, some platforms (e.g., iOS, Windows Phone, and Blackberry) show permission requests to the user at runtime, when an app first requests a resource. While runtime requests provide context to the user, a decade of usability research has shown that users may become habituated to these warnings, making them ineffective [2,6,8,23,25]. Because habituation increases with each warning exposure [15], researchers have studied ways to minimize unnecessary warnings [24] and to make important warnings more salient [5].

Capturing users' app privacy preferences is challenging [4]. Consolvo *et al.* showed that smartphone users value *why* requesters want access and the amount of detail given in the requests when deciding whether to disclose their locations [7]. Kelley *et al.* showed that users feel more comfortable sharing information when given granular privacy settings than with simple opt-in/opt-out mechanisms [14]. Sadeh *et al.* evaluated interfaces to help users better specify their privacy preferences by specifying rules and auditing disclosures [22]. Experiments by Fisher *et al.* build on this by showing that users do make decisions about which of their smartphone apps they allow to access their locations [13]. They surveyed iOS users and concluded that while users grant the majority of location requests, most users deny at least one app access to location. Felt *et al.* reported similar results: a majority claimed to have declined installing an app because of its permissions [12].

Lin *et al.* studied users' mental models of app behaviors [18]. They found that even when users read permission requests and are aware of the resources an app uses, users still have trouble understanding why certain resources are needed, which makes it difficult for users to make informed decisions based solely upon the permissions an app accesses. They also found that participants were significantly more comfortable when the purpose of resource requests was explained.

App	Resource	Purpose string
Gmail	Contacts	"Let Gmail use your contacts to autocomplete email addresses."
Instagram	Contacts	"In order to find your friends, we need to send address book information to Instagram's servers using a secure connection."
Instagram	Photos	"This allows Instagram to share photos from your library and save photos to your camera roll."
Scout	Calendar	"Let Scout check for entries with location info to help you get to appointments on time."
Scout	Contacts	"Easily drive to, search for, and share ETA with the people who matter most to you."
Snapchat	Contacts	"Your contacts will be transmitted to our servers and used to find your friends."
SPUN	Location	"because we want to give you news about where you are."
Transit	Location	"Your location will be used to fetch nearby stops and routes."
Twitter	Contacts	"Have more fun with your friends on Twitter."
Twitter	Location	"Enjoy the Twitter experience tailored to your current location."
Flashlight	Location	"For Compass Mini Map to work properly, location services need to be enabled."
RedLaser	Location	"RedLaser will use your location to search for product pricing and availability at nearby stores."
MapMyRun	Location	"To track your workouts please press OK."
Vine	Location	"This allows you to view and tag posts with locations around you."
Expedia	Location	"Your current location will be used to find airports and hotels near you."

Table 1. Pool of real apps and their requested permissions, as well as the purpose strings that participants saw during the *Purpose* task.

In this paper, we examine developer-specified explanations for resource requests. To the best of our knowledge, we are the first to examine how this iOS feature influences user behavior, the extent of its adoption, and the reasons for why developers choose to include or exclude these explanations.

USER BEHAVIOR

We performed an online experiment to determine whether users are more likely to grant iOS 6 permission requests when developer-specified explanations (i.e., "purpose strings") are present, whether users better understand the reasons for these requests when purpose strings are present, and whether users are more satisfied with their decisions.

Methodology

Our experiment involved three tasks. In the first two tasks, participants viewed a screenshot of a real app with a purpose string (the *Purpose* task) and a screenshot of a different real app with its purpose string removed (the *Control* task). We randomized the order of these two tasks. Each screenshot was randomly selected from a pool of 15 permission request screenshots (Table 1), taken from real apps and using real purpose strings. We included logic so that no one saw screenshots from the same app for both tasks. Thus, these two tasks allowed us to perform a within-subjects analysis of participants' decision-making with and without purpose strings.

1. *It helps me make a more effective decision about the sharing of my information.*
2. *It is useful.*
3. *It gives me more control over the sharing of my information.*
4. *It makes it easier to decide on the sharing of my information.*
5. *It allows me to quickly decide on whether to share my information.*
6. *It allows me to efficiently decide on whether to share my information.*
7. *It addresses my concerns over the sharing of my information.*
8. *I am satisfied with it.*
9. *It makes me more comfortable deciding on whether to share my information.*
10. *It is clear and easy to understand.*
11. *The language used is simple and natural.*
12. *I feel like it is necessary for me to make an effective decision about the sharing of my information.*

Table 2. Each participant answered 12 questions on a 7-point Likert scale ("strongly agree" to "strongly disagree"). We took the average of these 12 responses to create a "satisfaction score."

For the third task, we showed a screenshot of a fake app, *Party Planner*, requesting access to the user's contact list. This request featured a purpose string randomly selected from a pool of 14 (Table 4). This task allowed us to perform a between-subjects analysis of the effect of text choice in the purpose strings, while also controlling for app familiarity (i.e., no participant had previously interacted with our fictitious app).

For each of the three tasks, we included three pages of questions. On the first page, we displayed a screenshot of the app followed by a detailed description of what the app does. We then asked participants to type the name of the app and to report whether or not they had previously used the app.

Each task's second page showed a screenshot of one of the aforementioned permission requests. When participants viewed the *Purpose* task, the request featured a purpose string from Table 1; when they viewed the *Control* task, no purpose string was displayed; and when they viewed the *Party Planner* task, the purpose string was drawn at random from Table 4. We asked, "if you click 'OK,' what information will the app get access to?" They could select from ten multiple-choice options, including "none of the above" and "I don't know." We used this question to establish whether they had studied the screenshot. Next, we asked three open-ended questions: "Why do you believe the app is asking for access to this information?", "Do you believe that the app will use this information for any other purposes? If so, what?", and "Is there any additional information you would like to be shown in the dialog box? If so, what?" Two coders independently tagged responses to look for common themes. Finally, we asked whether they would allow the request.

On the third page of each task, participants rated 12 statements about the permission request they had just viewed on a 7-point Likert scale (Table 2). For each task, we averaged each participant's responses to all 12 statements, to create a "satisfaction score." While each statement was phrased positively, this is not a cause of bias because we only used these to make relative comparisons between experimental conditions.

Our survey concluded with demographic questions, including whether participants were iOS users. We recruited them through Amazon's Mechanical Turk, restricting participation to smartphone users based in the U.S. who were 18 or older. The entire experiment took approximately 15 minutes to complete and we compensated participants $2.50 for their time.

		User Decision		
Request Type	n	Allow	Deny	% Allow
Purpose Task	568	418	150	73.6%
Control Task	568	374	194	65.8%

Table 3. Participants' decisions (within-subjects) to allow or deny a request based on whether or not it included a purpose string.

Analysis

From the 826 responses that we received, we filtered out participants who claimed not to own a smartphone, specified a make/model that did not match a specified platform,[1] or wrote gibberish for the open-ended questions. This left us with 772 responses. Overall, our sample was skewed male (65.5% of 772), with an average age of 28 ($\sigma = 8.1$, ranging from 18 to 62), and 35.0% were iPhone users (58.9% used Android).

For each of the three tasks, participants named the app (open-ended) and its requested resource (multiple-choice). Due to some ambiguity, many chose a superset of the correct resource (e.g., they saw Twitter's location request, but from their familiarity with the app, they indicated that it would also request their contacts). We therefore accepted supersets of the correct responses. Based on the low reliability of these questions (Cronbach's $\alpha = 0.64$; i.e., getting one wrong was not a predictor of getting others in subsequent tasks wrong), we chose to filter participants out on a task-by-task basis. For example, we removed those who could not name both the *Purpose* task app and its requested resource from the *Purpose* task analyses, but included them in the *Party Planner* task analyses if they could correctly name that app and its requested resource. This reduced the number of participants to 666 for the Party Planner task and 623 for the *Purpose* and *Control* tasks. A technical error also caused some participants to be shown incorrect images in the *Purpose* and *Control* tasks, resulting in 568 participants in these conditions.[2]

Effect of Purpose Strings on Granting of Requests

When participants viewed purpose strings (i.e., the *Purpose* task), they approved 73.6% of the requests, whereas when the requests did not have purpose strings (i.e., the *Control* task), they approved 65.8% of the requests (Table 3). A Wilcoxon Signed Ranks test found this difference to be statistically significant ($Z = -3.569, p < 0.0005$).

Examining Explanatory Factors

To determine whether the significant change in behavior was due solely to the presence of the purpose strings or some other factor, we examined whether there were any observable effects due to the ordering of the within-subjects tasks, participants' familiarity with the specific apps in the first two tasks, or their familiarity with iOS in general.

[1]We asked participants what platform they used at the beginning of the survey (e.g., iPhone, Android, etc.) and then asked them to specify the make/model in the demographics section at the end.

[2]Entirely removing these participants does not change our findings.

We first examined the effect of the within-subjects questions' order (i.e., whether participants were more likely to allow a request based on whether they saw the *Purpose* or *Control* task first). We performed Fisher's exact test to compare the approval rates in the *Purpose* task (i.e., real apps with purpose strings) between participants who saw this task first and those who saw it after completing the *Control* task. We observed no statistically significant differences ($p < 0.481$). Likewise, we performed the same test on the *Control* task approval rates and also observed no statistically significant differences ($p < 0.244$). This suggests that the difference in approval rates was not due to task ordering or that participants deduced our goals and acted to please us: participants had no chance to deduce the stimulus until the second task (noticing that it added or removed a purpose string), leaving their first task untainted by demand characteristics. If participants were acting to please us, those who performed the *Control* task first (unaware of goal) would have behaved differently than participants who performed this task second (aware of goal).

We randomly displayed apps drawn from the same pool in the *Purpose* and *Control* tasks to prevent participants from being more familiar with the app in one task than the other. However, we also explicitly tested for this: we performed a within-subjects comparison to see if participants were more likely to report using the apps shown in requests containing purpose strings (25.5%) than the apps shown in requests not containing purpose strings (26.4%). A Wilcoxon Signed Ranks test found this difference to not be statistically significant ($Z = -0.333$, $p < 0.739$); nor did we observe correlations between allowing a request and app familiarity (*Control*: $\phi = 0.052$, $p < 0.217$; *Purpose*: $\phi = 0.012$, $p < 0.778$). Thus, participants' familiarity with the apps did not have an observable effect on their behavior.

We explored the similar question of whether participants' iOS familiarity affected decisions to allow permission requests. Neither of the first two tasks showed a significant difference in approval rates between current users of iOS and those of other platforms (Fisher's exact test; *Purpose*: $p < 0.769$; *Control*: $p < 0.023$; Bonferroni corrected $\alpha = 0.01$). Thus, we conclude that the presence of purpose strings was responsible for the significant difference in behavior.

User Satisfaction and Comprehension

In order to measure participants' level of satisfaction with each request, we asked them to rate 12 statements (Table 2) using a 7-point Likert scale ("strongly agree" to "strongly disagree"). For each task, we calculated participants' *satisfaction scores* as the average of all 12 questions. We then performed a within-subjects comparison of the satisfaction scores during the *Purpose* task ($\mu = 5.42$) and the *Control* task ($\mu = 4.97$): we observed a statistically significant difference (Wilcoxon Signed Ranks test; $Z = -8.053$, $p < 0.0005$). This indicates that participants felt more positively about their experiences when they were presented with requests containing purpose strings.

To examine why participants felt better about their decisions when presented with purpose strings, we examined the responses to our three open-ended questions. First, we observed that when viewing purpose strings, they were significantly less likely to say that they needed additional information to make a decision: 57.5% said they did not need additional information during the *Purpose* task versus only 43.6% during the *Control* task (Wilcoxon Signed Ranks test; $Z = -6.077$, $p < 0.0005$). Surprisingly, purpose strings did not observably help participants explain why their data was being requested: 6.0% could not provide an explanation for why their data was being requested during the *Control* task versus 5.4% during the *Purpose* task (Wilcoxon Signed Ranks test; $Z = -0.438$, $p < 0.662$).[3] Likewise, we discovered that the purpose strings were unlikely to better communicate whether personal data would be used for other purposes: 31.0% were uncertain during the *Purpose* task versus 32.7% during the *Control* task, which was not statistically significant (Wilcoxon Signed Ranks test; $Z = -0.873$, $p < 0.383$).

Thus, our data shows that while participants were significantly more likely to allow requests when presented with purpose strings and that they felt positively about these requests, the purpose strings did not help them understand why their data was being requested or how it might be used.

Choice of Text

We examined the *Party Planner* task data to better understand how the specific text of a purpose string may influence participants. The fictitious Party Planner's purpose strings were randomly drawn from Table 4. This allowed us to perform between-subjects comparisons to examine the effects of the specific text, while controlling for the app. We designed these purpose strings to convey varying amounts of information. "Contact access is required for this app to work properly" was used as a control statement because it contains little information, whereas the other purpose strings contain varying amounts of information to explain why data is being requested, how it will be used or stored, and whether it will be used for other purposes. When viewing the control statement, participants' satisfaction scores were significantly lower than they were when viewing the other statements (4.75 vs. 5.34; Mann-Whitney U test; $U = 13,377.0$, $p < 0.001$). Though despite this difference, we ultimately observed no differences in behavior: Fisher's exact test did not yield any statistically significant differences between the approval rates of participants shown the control purpose string and those shown the other purpose strings ($p < 0.495$).

ADOPTION

In the previous section, we showed that developer-specified explanations for permission requests impact user behavior and satisfaction. In this section we examine the adoption of this feature by developers. We performed a combination of automated and manual analysis to identify apps that requested access to any of the six resources that require user approval, and of these, what percentage included developer-specified explanations for resource requests (i.e., "purpose strings"). Finally, we examine trends in developers' explanations.

[3]This question measured participants' self-confidence on understanding the reasons for the request; we measured whether participants could provide any reason, regardless of its correctness.

Purpose String	Approval Rate
Control: "Contact access is required for this app to work properly."	52.5% of 59
"Let Party Planner use your contacts to autocomplete email addresses."	70.2% of 47
"To find friends, we'll need to upload your contacts to Party Planner. Don't worry, we're not storing them."	69.5% of 59
"Party Planner would like to access your address book to show you the cheapest attractions by your contacts' location. We won't use your contact information for any other purposes."	66.7% of 48
"Your contacts will be used to find your friends."	65.5% of 58
"In order to find your friends, we need to send address book information to Party Planner's servers."	62.5% of 48
"Have more fun with your friends on Party Planner."	58.7% of 46
"Easily search for and share event information with the people who matter most to you."	57.5% of 40
"Your contacts will be uploaded to our secure server. This data is maintained securely and is not shared with another party."	52.9% of 34
"Your contacts will be used to find your friends. They won't leave your phone."	51.5% of 33
"In order to find your friends, we need to send address book information to Party Planner's servers using a secure connection."	51.0% of 51
"Your contacts will be transmitted to our servers and used to find your friends."	46.2% of 39
"Party Planner would like to access your address book to show you the cheapest attractions by your contacts' location."	45.5% of 55
"Party Planner would like to access your address book to show you the cheapest attractions by your contacts' location and other purposes."	38.8% of 49
Total:	56.8% of 666

Table 4. Pool of app purpose strings for the fictitious Party Planner app, as well as their associated approval rates. The first purpose string was used as a control condition because it conveys no information about why the app is requesting access.

Methodology

We created a corpus of 4,400 free iOS 6 apps by downloading the top 200 apps for each of the 22 app categories in the iTunes Store.[4] Estimating the total adoption rate across our entire corpus took three steps. First, we calculated the numerator by extracting purpose strings from each app's plaintext metadata file. Next, we calculated the denominator by performing static analysis on the decrypted binaries to estimate how many apps requested access to any of the six protected resources (i.e., including those without purpose strings). Finally, we manually examined 140 apps to validate our results.

Purpose String Extraction

Applications for iOS have two parts, an encrypted binary (i.e., the executable, which is decrypted at runtime) and unencrypted metadata files. Purpose strings are optionally included in the latter (i.e., `Info.plist` or `InfoStrings.plist`), known as a "property list" or `plist`. We searched for these files within each app in our corpus, recording any purpose strings that they contained.

In previous versions of iOS, developers had the option of specifying a purpose string for location requests (though not for other resources). Because the purpose string was passed as a parameter to API calls in older versions, they are stored encrypted and cannot easily be extracted (unlike when the purpose strings are stored in `plist` files). For backwards compatibility, iOS 6 still displays purpose strings specified via this deprecated method, which means that only extracting purpose strings from `plist` files will not yield all the possible purpose strings that a user is likely to encounter. We used static analysis of the binaries to close this gap.

Static Binary Analysis

To measure adoption of purpose strings, we counted the number of apps for which developers had the opportunity to specify purpose strings. We performed static analysis on our corpus of 4,400 apps to determine the number of apps that requested access to any of the six protected resources.

[4]We collected this data during May 15-16, 2013.

iOS app binaries obtained from the iTunes App Store are stored on the user's device in encrypted form. We decrypted the 4,400 apps by loading them onto a jailbroken iPhone and used the iOS app decryption tool, *Rasticrac* [19]. We used iOS 6.0.2 on the iPhone, which was the latest version for which a jailbreak method was available. Any apps that required a newer version of iOS were excluded and replaced with another app ranked in the top 200 for the same category.[5]

After decryption, we used the Unix `strings` tool to extract function names. By searching for the names of the API functions used to access the six protected resources, we could automatically determine whether a given app would access any of those resources. Additionally, we also searched for the method that is invoked when developers specify a location purpose string using the deprecated API method. Although we could not obtain the actual deprecated location purpose strings using the `strings` tool alone, this method allowed us to gain a more accurate understanding of total adoption.

Manual Testing

We were concerned that simply searching for function names using our static analysis method might yield false positives. For instance, if a request to access a user's contacts is contained within a section of unreachable code, our static analysis would indicate that this app requests access to contacts, whereas in reality no such request would ever be made. Similarly, if the set of function names we searched for was incomplete, our static analysis would report false negatives. Therefore, we validated our static analysis results by performing manual testing on a subset of apps in our corpus.

We selected the top 10 free apps in 14 categories, resulting in 140 apps. We used manual testing to record all resource requests made by these 140 apps, and their associated purpose strings, if any. If a purpose string was provided for a location resource request, but no corresponding string was found in its `plist` files, we deduced that this purpose string was specified using the deprecated API method.

[5]The iTunes top 200 apps by category frequently changed, and so we simply had to wait for replacement apps to appear.

	# Apps With		
Resource	Purpose Strings	Resource Requests	Adoption Rate
Reminders	6	82	7.3%
Contacts	66	1,200	5.5%
Location	57	2,539	2.2%
Calendars	11	508	2.2%
Photo & Video	35	2,631	1.3%
Location (Deprecated)	548	2,539	21.6%
Any	660	3,486	18.9%

Table 5. Adoption rates by resource. Many apps were predicted to request multiple resources and some apps provided purpose strings for multiple resources. The row labeled *Location (Deprecated)* lists the purpose strings using the deprecated API method.

Analysis

Overall, of our corpus of 4,400 apps, we estimated that 3,486 (79.2%) request access to one or more of the six protected resources. Of these, 660 apps (18.9% of 3,486) included developer-specified explanations (i.e., "purpose strings"). Our manual analysis of 140 apps corroborated these findings: 17.4% (16 of 92 apps that requested access to protected resources) included purpose strings.

Adoption Rates

We combined our `plist`-specified purpose string data with our static analysis predictions to determine the overall adoption rate for developer-specified purpose strings in resource requests. Table 5 depicts our results, classified by resource.[6]

Of the 140 apps that we tested manually, 92 requested access to at least one of the six protected resources. We found that 16 apps (17.4% of 92; 95% CI: $[10\%, 27\%]$) included purpose strings in these requests. Manual testing found no false negatives in our static analysis: static analysis found every resource request that we found through manual testing. Our static analysis predicted that 130 of the 140 apps accessed protected resources, but we were only able to confirm this using manual testing for 92 apps. This indicates a false positive rate of up to 41%, which further suggests that the overall purpose string adoption rate is likely higher than estimated in Table 5 (because the denominator—the total number of apps requesting access to protected resources—is likely smaller). Data collected by Agarwal *et al.* support this hypothesis; the percentages of 222,685 apps in their study found to access the location and contacts resources were roughly one fourth of those estimated by our static analysis [1].

Ad libraries may account for the large false positive rate. Pearce *et al.* show that 46% of ad-supported Android apps request permissions used only by ads [21]. We examined the apps yielding false positives for evidence of 6 popular ad libraries and found evidence in 71% (27 of 38). When app developers include an ad library but do not target ads, targeting methods appear as dead code in the app binaries. Our static analysis would then predict a resource request that disagrees with manual testing results, inflating the false positive rate.

[6]Bluetooth has been excluded since only three apps in our corpus requested this resource, none of which included a purpose string.

Approximately 21.6% (548 of 2,539) of apps requesting location data use the deprecated API to specify a purpose string, whereas only 2.2% (57 of 2,539) use the new method. Across all resources, 3.6% (125 of 3,486) of apps specify a purpose string using the new method. This suggests that it is only a matter of time before developers use the new method to communicate to users why their apps request these resources.

Purpose String Categories

We extracted 175 unique purpose strings specified in the `plist` files of 125 apps. Multiple people each read through all of the strings and then created a set of categories based on the themes that they observed. Next, two coders independently tagged each purpose string with one or more of the following agreed-upon categories:

- *User benefit*. The purpose string describes how granting permission to the request will benefit the user's experience.
  - "This allows Instagram to share photos from your library and save photos to your camera roll."
  - "Have more fun with your friends on Twitter."
- *Appropriate resource use*. The purpose string promises or guarantees that the accessed information will not be misused, or that it will be used only for the described purposes.
  - "We'll include your old photos in Timehop! (Don't worry, photos won't be uploaded or shared)"
  - "So you can add them more easily. We'll never misuse them, promise!"
- *Local/Non-local use*. The purpose string explains whether the accessed information will be used locally or uploaded.
  - "To find family and friends, Voxer needs to send your contacts to our server."
  - "To import photos to hidden albums or browse your visible albums, CoverMe needs to access your camera rolls. Your photos won't be uploaded to CoverMe servers."
- *Security*. The purpose string describes how the accessed information will be protected against unauthorized access.
  - "Securely sync your contacts so we can help you find more friends."
  - "AllTrails can securely and anonymously check your contacts to see who else is using the app"
- *Information sharing*. The purpose string describes the policy for sharing information with third parties.
  - "We use your location to help you find nearby dishes and restaurants. We won't ever share it."
  - "Call Bliss requires contact access so you can allow callers. Your contacts remain confidential, Call Bliss does not copy or share them."
- *Data storage*. The purpose string explains whether the data will or will not be stored, or addresses the storage duration.
  - "WhatsApp copies your address book phone numbers to WhatsApp servers to help you connect with other WhatsApp users"
  - "We take your privacy seriously and will never store or share your contacts."

Category	# Strings	% Strings
User benefit	172	98.3%
Appropriate resource use	24	13.7%
Local/Non-local use	15	8.6%
Security	8	4.6%
Information sharing	6	3.4%
Data storage	5	2.9%

Table 6. Categories of purpose strings found in our corpus. The categories were not mutually-exclusive.

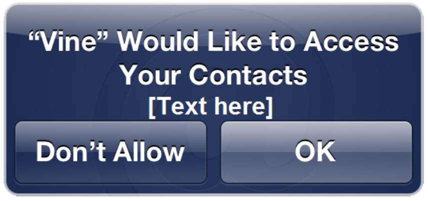

“Vine” Would Like to Access Your Contacts
[Text here]

Figure 2. Participants created purpose strings for two different requests.

Table 6 depicts the number of purpose strings that fell into each category. Nearly all strings (98.3% of 175) mentioned the benefits of granting access to the requested resource. The second most common category was assurances against inappropriate resource use, with 13.7% of strings providing this information. The categories were not mutually-exclusive (i.e., many purpose strings fell into multiple categories).

DEVELOPER OPINIONS

We surveyed iOS developers to determine if they were aware of purpose strings, the reasons why they were or were not using them, and the information they might include in them.

Methodology

To screen participants, we asked them how many iOS apps they had previously developed and whether any were available in the iTunes store. Following this page, we provided participants with an explanation and screenshot of a “purpose string” and asked them if they had heard of this feature or used it in any apps that they had developed. We asked them to select all the resources that their apps utilize from a multiple-choice list, which included the six protected resources. If applicable, we asked them to specify all the resources for which they had written purpose strings. Based on participants’ previous responses, we asked them why they did or did not include purpose strings in their apps. Multiple coders categorized the open-ended responses to determine the reasons why developers choose to use purpose strings.

We examined the types of purpose strings developers write by presenting participants with descriptions and screenshots of two popular apps, Vine and Scout. We subsequently displayed screenshots of each app’s request for a user’s contacts, using a placeholder for a purpose string (Figure 2); we asked participants to write purpose strings for each of these requests. Each app description included an explanation for how the data would be used, adapted from each app’s real privacy policy. The goal of the task was to observe how developers would transform these descriptions into purpose strings:

Total developers surveyed							30	100%
Access a protected resource					28	93.3%		
Aware of purpose strings			15	53.6%				
Use purpose strings	7	46.7%						
Do not use purpose strings	8	53.3%						
Unaware of purpose strings			13	46.4%				

Table 7. Developers’ use and awareness of purpose strings. For instance, of the 28 developers who have written apps that request protected resources, thirteen (46.4% of 28) were unaware of purpose strings.

- **Vine:** *Our Services include several features to help you find the accounts of people you already know. For example, you may upload information from your address book or connect your Vine account to your account on another service such as Twitter. Vine will match the information you provide with the contact information of other Vine users. We do not retain your address book information after displaying these matches.*
- **Scout:** *You can allow our application to access the data stored in the address book and/or calendar on your mobile device. Our applications may use data from your address book to allow you to search for and navigate to the addresses of your contacts and our applications may use calendar entry data to provide you with reminders about upcoming appointments and to provide navigation to the locations of your appointments. We will only access this data if you have given permissions for a particular Telenav application to do so, and you can revoke such permissions at any time by changing the settings on your mobile device.*

Our survey concluded with open-ended questions to help us confirm participants were iOS developers: the programming languages used, the testing/debugging tools used, the number of years developing iOS apps, and the target iOS versions.

We recruited participants through ads on StackOverflow[7] and iPhoneDevSDK.[8] On StackOverflow, we placed our advertisement on pages with tags related to iOS (“ios,” “ios6,” “ios-simulator,” or “ios-maps”), since these pages were likely to be visited by iOS developers. Participants who completed the survey were entered into a raffle for one of four $100 Amazon gift cards. We required participants to have developed at least one app that was approved for the iTunes App Store.

Analysis

Our survey was visited by 102 people, with 74 recruited from StackOverflow and 14 from iPhoneDevSDK.[9] Of these, 53 people opted to not participate and 16 stated that they had not developed iOS apps and were disqualified. Among the 33 participants who submitted responses meeting our requirements, three could not answer questions about the tools they use to develop iOS apps and were removed. This left us with 30 valid responses (91% of 33).

Developer Awareness

The results suggest that the vast majority of developers could be using purpose strings: 28 developers (93.3% of 30) claimed that the apps they currently have available in the

[7]http://stackoverflow.com/
[8]http://iphonedevsdk.com/
[9]14 responses did not specify the referrer.

iTunes Store access at least one of the six protected resources (Table 7). However, only 17 developers (56.7% of 30) in our survey claimed to be aware of purpose strings. Of the developers who had requested a protected resource and were aware of purpose strings, seven (46.7% of 15) reported having used a purpose string; the two developers who did not access protected resources were nonetheless aware of purpose strings.

We found no relationship between developers' awareness of purpose strings and their years of experience developing iOS apps nor the number of apps that they had developed, which may suggest that lack of awareness is due to poor documentation of the feature, rather than developer inexperience. In the remainder of this section we explore developers' responses to the open-ended questions, including why they chose to use purpose strings (or not) and the themes that emerged from the example purpose strings that they wrote.

Developer Attitudes

We asked the eight participants who claimed to not use purpose strings, despite being aware of them and writing apps that accessed protected resources, why they chose not to use them. Two participants omitted responses to this question. Two indicated that they intentionally do not use them because they use separate messages within their apps to inform users:

- "We try to give the user a heads-up before that permission dialog appears"
- "Because we include an entire screen explaining the purpose ahead of requesting it."

Similarly, the remaining four thought they were unnecessary:

- "Don't think it's necessary."
- "Because the default description is good enough."
- "In my case it seems obvious, but I will include a purpose string in the next update."
- "I haven't had to."

All seven of the participants who use purpose strings indicated that they believed they are effective at communicating why a resource is necessary. Examples included:

- "To make it clear to the user why a permission is being requested, especially for non-obvious cases."
- "To inform the user of why we are asking for permission."

To gain insight into the types of information developers find important in purpose strings, we examined the strings that they provided for the Vine and Scout resource requests. Multiple coders categorized these strings according to the same categories and methodology in the Adoption section (Table 8). Every single purpose string explained why data was being requested (two participants omitted Vine purpose strings, while four omitted Scout ones). Examples included:

- "Contacts are used to find the people you want to share your videos with."
- "Granting permissions to your contacts will allow Scout to search for and navigate to the addresses of your contacts."

For Vine, it was common to also discuss the storage of users' data. We suspect that this was because Vine's privacy policy (and therefore our description) included information about how data will be used and stored, whereas Scout's did not. Some examples of these purpose strings included:

	Vine		Scout	
Category	# Strings	% Strings	# Strings	% Strings
User benefit	28	100%	26	100%
Data storage	7	25.0%	0	0%
Local /Non-local use	5	17.9%	0	0%
Appropriate use	3	10.7%	0	0%
Security	1	3.6%	0	0%
Information sharing	0	0%	0	0%

Table 8. Classification of developer-written purpose strings. Some developer-written purpose strings fell into more than category.

- "To quickly find your friends on vine and give you a better experience, we need access to your contacts. All information is safely sent and stored."
- "Vine uses your contacts to let you share videos. We never contact anyone without your permission, which can be revoked at any time."

The results of our developer survey suggest that the main reason developers do not use purpose strings is simply due to lack of awareness, which also explains why we observed so many deprecated API functions (see Adoption section). That said, a majority of developers believe that it is important for users to understand why their information is being requested, even if they choose to notify them through in-app methods rather than with purpose strings.

DISCUSSION

In this section, we discuss what our findings mean in terms of the impact of purpose strings on users and how developers might be encouraged to make privacy information more apparent to users. We conclude with the limitations of our experiments and future work.

Influence on User Understanding and Behavior

Although users grant resource requests with purpose strings more often than those without, there does not appear to be much change in their behavior as a function of the specific text used within a purpose string. In our user survey, requests with purpose strings were granted or denied independently of whether the purpose string contained useful information.

A possible explanation for this effect may be similar to what Langer *et al.* found in experiments where people asked to cut in line while making photocopies: people were significantly more amenable when the requester provided any explanation, regardless of whether or not that explanation was actually relevant to the request [16]. In the case of purpose strings, simply having something that looks like an explanation may make users more compliant with the request. Another possibility is that the inclusion of a purpose string distorts the look and feel of the request dialog enough to break habituation; that is, it might not matter what the request says, simply that it looks different from previous requests. However, if this were the case, we would expect to see a correlation with whether users were existing iOS users, which we did not.

Improving Developer Utilization

Despite not observing any effects on behavior based on the choice of text, we observed through our adoption and developer experiments that the vast majority of developers try to provide users with explanations that address why information is being requested. Unfortunately, users are presented with purpose strings in less than one out of five apps.

The results of our developer survey suggest that adoption may be low partly because developers are unaware of purpose strings or how to use them. Furthermore, developer awareness of purpose strings did not correlate with the number of years of iOS development experience nor the number of iOS apps developed. This suggests that the low adoption rate is not simply a reflection of developer inexperience, but a result of poor documentation. Anecdotally, we found the iOS documentation incomplete and sometimes contradictory during the course of our experiments. Creating developer resources to improve app privacy may increase developer compliance.

A consequence of allowing developers full freedom in specifying purpose strings is that many fail to effectively utilize them. An alternate approach would be to provide developers with a set of pre-defined purpose string templates from which they can choose. These templates could address the types of information we commonly found in our collected purpose strings, such that the context of the requests can be made clear to users [20]. These templates might also make it easier for developers to supply purpose strings. Of the 23 developers we surveyed who were either unaware of purpose strings or did not use them, thirteen said that they would be "very likely" to use pre-defined templates, if given the option in the future.

Experimental Limitations

We used a simple form of static analysis to predict resource requests and purpose strings specified using the deprecated API method. Due to its simplicity, this analysis does not consider many of the factors necessary to provide more accurate predictions (e.g., by performing dynamic analysis).

Although we provide insight on users' understanding of resource requests, we rely on users to self-report this understanding. While our user study was controlled so that we could observe relative differences as a function of the presence of purpose strings, it is not clear how these findings might translate to the real world where users are influenced by other factors; we measured the efficacy of purpose strings in altering behavior, rather than their effectiveness. Likewise, we measured users' willingness to approve requests, rather than whether or not they were acting in their best interests. Despite not risking the disclosure of any real data, the significant differences we observed between randomly-assigned conditions show that participants took the tasks seriously.

Our experiments were limited to free iOS apps. Although we expect similar results for paid apps, an exception may be in the developer adoption rate of purpose strings. Paid apps are less likely to contain advertising libraries [21], which could reduce the number of resources requested by apps, thus increasing the estimated adoption rate.

The primary purpose of this study was to examine developer adoption of purpose strings and their impact on user behavior. Since we controlled for the apps participants viewed (comparing popular apps with the same popular apps, and an unknown app with the same unknown app), future work is needed to examine how the trustworthiness of an app influences a participants' willingness to grant permission requests. While we observed significant differences in user behavior across the resources we tested, additional research is needed to examine whether behaviors change based on the resource accessed. Such studies should be performed in situ, such that users understand their real data is at risk.

ACKNOWLEDGMENTS

This work was supported in part by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422), NSF award number CNS-1318680, and by Intel through the ISTC for Secure Computing. Thanks to Rebecca Pottenger, Erika Chin, and Aimée Tabor for their support and feedback.

REFERENCES

1. Agarwal, Y., and Hall, M. ProtectMyPrivacy: detecting and mitigating privacy leaks on iOS devices using crowdsourcing. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, MobiSys '13, ACM (New York, NY, USA, 2013), 97–110.

2. Amer, T. S., and Maris, J. B. Signal words and signal icons in application control and information technology exception messages – hazard matching and habituation effects. Tech. Rep. Working Paper Series–06-05, Northern Arizona University, Flagstaff, AZ, October 2006. `http://www.cba.nau.edu/Faculty/Intellectual/workingpapers/pdf/Amer_JIS.pdf`.

3. Apple Inc. What's New in iOS. `https://developer.apple.com/library/ios/releasenotes/General/WhatsNewIniOS/Articles/iOS6.html`, January 28 2013. Accessed: September 15, 2013.

4. Benisch, M., Kelley, P. G., Sadeh, N., and Cranor, L. F. Capturing location-privacy preferences: quantifying accuracy and user-burden tradeoffs. *Personal Ubiquitous Comput. 15*, 7 (Oct. 2011), 679–694.

5. Bravo-Lillo, C., Komanduri, S., Cranor, L. F., Reeder, R. W., Sleeper, M., Downs, J., and Schechter, S. Your attention please: designing security-decision UIs to make genuine risks harder to ignore. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, ACM (2013), 6.

6. Brustoloni, J., and Villamarín-Salomón, R. Improving Security Decisions with Polymorphic and Audited Dialogs. In *Proceedings of the 3rd Symposium on Usable Privacy and Security*, SOUPS '07, ACM (2007), 76–85.

7. Consolvo, S., Smith, I. E., Matthews, T., LaMarca, A., Tabert, J., and Powledge, P. Location disclosure to social

relations: why, when, & what people want to share. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, ACM (New York, NY, USA, 2005), 81–90.

8. Egelman, S., Cranor, L. F., and Hong, J. 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, ACM (New York, NY, USA, 2008), 1065–1074.

9. Enck, W., Gilbert, P., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., and Sheth, A. N. 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, USENIX Association (Berkeley, CA, USA, 2010), 1–6.

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

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

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

13. Fisher, D., Dorner, L., and Wagner, D. Short paper: Location privacy: User behavior in the field. In *Proceedings of the Second ACM workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '12, ACM (New York, NY, USA, 2012), 51–56.

14. Kelley, P. G., Benisch, M., Cranor, L. F., and Sadeh, N. When are users comfortable sharing locations with advertisers? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 2449–2452.

15. Kim, S., and Wogalter, M. Habituation, dishabituation, and recovery effects in visual warnings. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 53, SAGE Publications (2009), 1612–1616.

16. Langer, E., Blank, A., and Chanowitz, B. The Mindlessness of Ostensibly Thoughtful Action: The Role of "Placebic" Information in Interpersonal Interaction. *Journal of Personality and Social Psychology 36*, 6 (1978), 635–642.

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

18. Lin, J., Amini, S., Hong, J., Sadeh, N., Lindqvist, J., and Zhang, J. Expectation and purpose: Understanding users' mental models of mobile app privacy through crowdsourcing. In *Proceedings of the Second ACM workshop on Security and Privacy in Smartphones and Mobile Devices*, UbiComp '12, ACM (New York, NY, USA, 2012), 51–56.

19. Mongolo. Rasticrac v3.0.1. `http://iphonecake.com/bbs/viewthread.php?tid=106330&extra=page%3D1`, April 17 2013. Accessed: September 15, 2013.

20. Nissenbaum, H. Privacy as contextual integrity. *Washington Law Review 79* (February 2004), 119.

21. Pearce, P., Felt, A. P., Nunez, G., and Wagner, D. AdDroid: privilege separation for applications and advertisers in Android. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '12, ACM (New York, NY, USA, 2012), 71–72.

22. Sadeh, N., Hong, J., Cranor, L., Fette, I., Kelley, P., Prabaker, M., and Rao, J. Understanding and capturing people's privacy policies in a mobile social networking application. *Personal Ubiquitous Comput. 13*, 6 (Aug. 2009), 401–412.

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

24. Thompson, C., Johnson, M., Egelman, S., Wagner, D., and King, J. When it's better to ask forgiveness than get permission: attribution mechanisms for smartphone resources. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, ACM (2013), 1.

25. Xia, H., and Brustoloni, J. C. 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, ACM (New York, NY, USA, 2005), 489–498.

26. Yang, J. Smartphones in use surpass 1 billion, will double by 2015. `http://www.bloomberg.com/news/2012-10-17/smartphones-in-use-surpass-1-billion-will-double-by-2015.html`, 2012.