–Independent Work Report Spring 2014–

Clarity of Facebook Connect login permissions

Nicholas Robinson
Center for Information Technology Policy
Princeton University
ncrobins@princeton.edu
Advisor: Joseph Bonneau

May 6, 2014

This paper represents my own work in accordance with University regulations.

Nicholas Robinson

Abstract

We study Facebook Connect's permissions system to determine if it functions the way users and developers expect. Through web crawling, experimentation, surveys, and research, we determine that its behavior is unexpected in many areas. We show that it can create permission requests for more permissions than the developer intended and consequently permissions that allow a site to post to the user's profile are granted on an all-or-nothing basis. We discuss how the requested permissions are presented to the user and we demonstrate that users generally understand the data sites can see on their profile but do not understand the many different things the sites can post.

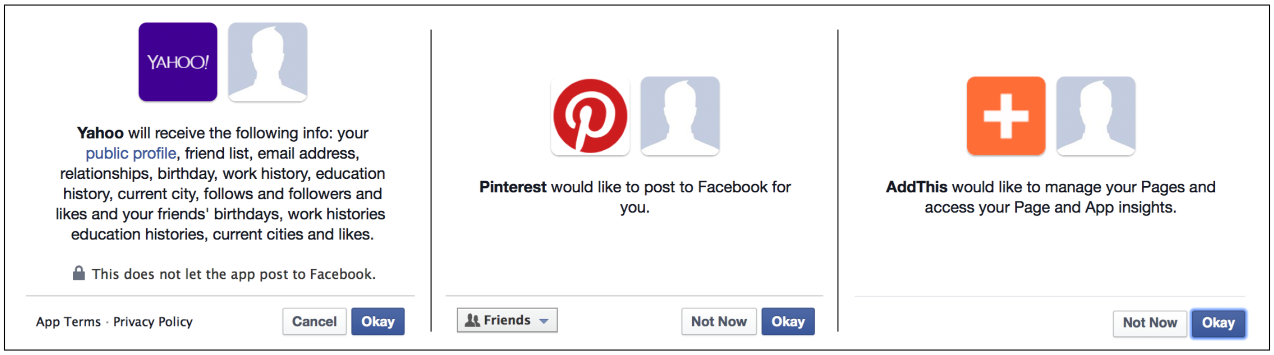
1. Introduction

Single Sign-On (SSO) systems allow users to log in to websites using their username and password from a third-party identity provider, such as Facebook, Google, or Twitter. This creates fewer passwords for users to remember, theoretically meaning that they can have more complicated and therefore more secure passwords [20]. SSO systems have become increasingly common; Facebook's service is perhaps the most common.

Facebook Connect does more than just allow a user to sign in: sites can request access to parts of the user's Facebook profile (this is a feature of many SSO systems based off of the OAuth protocol).

When the developer integrates the login system with their website, they design it to request various permissions from Facebook. These permissions allow them to see information on the user's profile or publish content to their profile.

When a user logs in to a website using Facebook Connect, they are presented with up to three dialogues asking them to authorize the website to receive the desired permissions. The first message corresponds to the *read* permissions, permissions that allow the site to see information on the user's profile. The second corresponds to the *write* permissions, which allow the site to post various things on the user's profile. The third (or second, if no write permissions were requested) corresponds to the *extended* permissions. These permissions are for more sensitive things, such as sending the user text messages and marking their notifications as read. The user can refuse to grant write and extended permissions and still log in.[1] If the user approves these permissions, they are logged in to the site and the site receives their information from Facebook. Figure 1 shows examples of messages asking the user to authorize permissions.



Yahoo will receive the following info: your public profile, friend list, email address, relationships, birthday, work history, education history, current city, follows and followers and likes and your friends' birthdays, work histories education histories, current cities and likes.

This does not let the app post to Facebook.

Pinterest would like to post to Facebook for you.

AddThis would like to manage your Pages and access your Page and App insights.

Figure 1: Examples of messages presented to the user. From left: Read permissions message from Yahoo.com, write permissions message from Pinterest.com, and extended permissions message from AddThis.com.

Users logging in with Facebook Connect place a lot of trust in Facebook to only share information that the user authorizes. This relies not only on Facebook granting only the permissions presented in the authorization messages but also on users correctly interpreting these messages. We will explore both in this paper and show that:

[1]The documentation [11] provides more details about these groups, although it breaks them down into six different groups.

- Facebook Connect sometimes asks the user to authorize more permissions than the developer intended to request.
- Write permissions are granted to sites on an all-or-nothing basis. For example, if a site can update the user's status, it can also upload photos.
- Users generally understand which read permissions are being requested when they log in, although many think that they are only giving access to data marked as public.
- Users do not understand the variety of write permissions sites will receive upon authorization.

2. Related research

Many researchers have studied the security and permissions systems of various apps[2] and SSO systems. Sun and Beznosov [21] uncovered vulnerabilities in many major OAuth SSO implementations. Chaabane et al. [3] and Huber et al. [18] identified information leaks in Facebook and RenRen apps. There have also been several studies of what permissions sites request, such as Frank et al.'s study in which Facebook apps were grouped into categories based on the permissions they request [15].

Some studies have tested user comprehension of SSO systems as well. A 2011 audit of Facebook Ireland looked at, among other things, how clearly the Facebook app system is presented to users. It also states that it "is not possible for an application to access personal data over and above that to which an individual gives their consent or enabled by the relevant settings"—that is, Facebook's permissions do appropriately limit what data an app can access [19].

Sun et al. studied user understanding of the authentication process in general—for example, whether users understood that the site they are logging in to cannot see the password for the identity provider (Facebook, Google, etc.) [22]. The study most directly related to ours is Egelman [6], which studied whether users were willing to use Facebook Connect and how well they understand (and how much they pay attention to) the permissions messages. Egelman concluded that 88% of users have a general understanding that their profile information will be shared with the site they

[2]The Facebook Connect SSO system uses the same system as native Facebook apps—creating a Facebook login on a website requires creating a Facebook app [8].

are logging in to, but that they typically do not pay attention to the specifics of the dialogues and do not make their decision whether to use Facebook Connect based on which permissions are being requested.

Our study differs from previous studies by determining what specific permissions correspond to the messages presented to the user and by evaluating user comprehension of these permissions. This lets us answer most precisely whether users understand exactly what information they are sharing by using Facebook Connect. In addition, Egelman only looked at read permissions. We found that write permissions are much more confusing to users.

3. Which permissions does Facebook actually give?

The first step to determining whether permissions are presented clearly is understanding which permissions are actually being granted. Facebook Connect's process of a site requesting permissions from a user can be broken down into three steps:

1. When the developer integrates their site with Facebook Connect, they create a login flow that requests a set of permissions from Facebook.
2. When a user logs in to the site using Facebook Connect, Facebook's API creates a request for a set of permissions.
3. Facebook presents the user with a dialogue asking them to approve a set of permissions.

Theoretically, these three sets of permissions should be the same: The developer asks for certain permissions, Facebook formats a request for those and only those permissions, and Facebook clearly identifies all of those permissions to the user. This section of the paper explores these three sets of permissions and analyzes their similarities and differences.

3.1. Methodology

Unfortunately, Facebook Connect's documentation [7] consists of a lot of incomplete and outdated information supplemented by blog posts from their engineers. As such, there is very little explanation of how permissions the developer requests are presented to the user for approval.

To gain a better understanding, we combined information from the documentation, observations from creating our own Facebook Connect login for a dummy site, and data from a couple hundred real sites.

3.1.1. Getting data from real sites The data from real sites was obtained by web crawling. The first step was to compile a list of sites with Facebook Connect logins. Starting with the most recent (October 2013) AppInspect [18] database of 25,000 Facebook apps, we filtered it down to only apps that had pages on the Facebook App Center that listed external websites (about 400). We then manually searched these to find which ones had Facebook Connect logins, which brought the list down to 91 sites.

Unfortunately, the AppInspect database does not include apps that are used solely for Facebook Connect—it only includes those that have native Facebook apps (although some of these also have sites with Facebook Connect). In part because of this, the majority of the sites we found from the database are relatively obscure. To make up for these deficiencies, we manually went through the Alexa Top 500 [2] websites from February 27th, 2014 and identified the ones with Facebook Connect logins (112 sites).

Combining the two lists yields 203 sites, about half of which receive a lot of monthly traffic (those from the Alexa Top 500) and about half of which do not (those from the AppInspect database). This list provides a diverse sampling of the websites people log in to with Facebook Connect.

Using a Selenium-based web crawler being developed by the Princeton Center for Information Technology Policy (CITP)'s Webfairness group, we automated logins to all 203 sites and recorded the permissions they request and the corresponding messages presented to the users. (The way the permissions they request was determined will be discussed in Section 3.3.1.) Twenty-six of the 203 sites used an older implementation of Facebook Connect; this paper will focus on the 177 with the current format.

3.2. How developers request permissions

The first step in the permissions presentation process as we outlined it above is the developer designing their site to request permissions from Facebook. For Facebook Connect, this is done by listing the desired permissions in a "scope" or "data-scope" parameter when the login process is initiated using Facebook's JavaScript SDK, Facebook's login button, or a manually built login system [11]. The developer can request any of the permissions listed in the documentation [10], although some are now deprecated (including, it appears, some that are not listed as deprecated).

The value of the scope is visible in the URL of the page where the user is asked to approve the read permissions, after the string "scope=" (see Figure 2). We confirmed using the test site that the values that show up in the URL are indeed the exact ones the developer puts in the site's scope parameter. When crawling the sites, we parsed the URLs to extract the permissions requested by developers.

```
https://www.facebook.com/dialog/oauth?app_id=138615416238413&client_id=138615416238413&display=
popup&domain=www.timecrunch.me&e2e=%7B%7D&locale=en_US&origin=1&redirect_uri=http%3A%2F
%2Fstatic.ak.facebook.com%2Fconnect%2Fxd_arbiter%2F8n77RrR4jg0.js%3Fversion%3D40%23cb%3Df1
7cdd7f881aff%26domain%3Dwww.timecrunch.me%26origin%3Dhttp%253A%252F%252Fwww.timecrunch
.me%252Ff2dc123d3ff5394%26relation%3Dopener%26frame%3Df25cadbc2ca73cc&response_type=token
%2Csigned_request&scope=email%2Ccreate_event%2Coffline_access%2Cuser_groups%2Cfriends_group
s%2Cpublish_stream&sdk=joey
```

Figure 2: The URL for the first message presented to the user when logging in to Timecrunch.me. The scope and permissions are colored red. The permissions being requested are *email*, *create_event*, *offline_access* (deprecated), *user_groups*, *friends_groups*, and *publish_stream*. ('%2C' represents a comma.)

3.3. How Facebook creates the permissions request

In the second step, a user visiting a site initiates the login process and Facebook creates a set of permissions that the site will get from Facebook if the user approves them. Theoretically and ideally this set would be identical to the one that the developer requested, but in practice this is not the case.

To begin, there are two permissions that are identified as "Basic Info/Default permissions" and are "included as part of every permissions request" [11]. These two permissions are *public_profile*, which gives access to the user's public profile, and *user_friends*, which gives access to the user's friends list. The documentation does not mention any other permissions that may be requested outside of what the developer asked for.

3.3.1. Identifying the actual permissions being requested The Facebook documentation does not say how a user presented with a message asking them to authorize permissions can identify the exact permissions being requested—they have only the message presented to them. But to understand how Facebook handles permissions, we need to know exactly which permissions will be granted if the user clicks okay.

The first page presented to the user when asking them to approve permissions has three hidden input HTML elements named *read*, *write*, and *extended* whose values are permissions (see Figure 3). Further investigation indicates that these three elements hold all of the permissions that will be granted if the user clicks okay (and the scope variable in the URL does not).

```
<input type="hidden" autocomplete="off" name="read"
value="email,user_groups,friends_groups,public_profile,user_friends,private" />

<input type="hidden" autocomplete="off" name="write"
value="publish_stream,publish_actions,create_note,photo_upload,publish_checkins,
share_item,status_update,video_upload" />

<input type="hidden" autocomplete="off" name="extended"
value="create_event,rsvp_event" />
```

Figure 3: From top: The *read*, *write*, and *extended* input elements from the HTML for the first message presented to the user when logging in to Timecrunch.me. Permissions are colored red. *private* is not actually a permission, but always appears in the *read* element. *offline_access* does not appear since it is deprecated. If a site did not have any permissions of one type, that element's value would be empty.

First, *public_profile* and *user_friends* (the aforementioned default permissions) always appear in the element named *read* although they never appear in the scope unless explicitly requested by the developer. We verified this both in the test site and with the data from the sites we crawled: All 177 sites with the current login format had both permissions in the *read* input element.

In addition, we identified several permissions that when requested by the developer cause certain other permissions to appear in the input elements with them. (These will be discussed in more detail in the Section 3.3.2.) With the test site we verified that the permissions the site gets are the ones specified in the HTML, not just the ones specified in the scope. For example, the Facebook documentation has an example of how to publish a story (liking an article) on the user's timeline using Facebook Connect [8]. It states that this requires the *publish_actions* permission. However, if we (as a developer) request the *create_note* permission in the scope, *publish_actions* appears in the HTML in the *write* input element and the story can still be published. (Requesting a permission such as *email* does not put *publish_actions* in the HTML and the story cannot be published.)

Finally, the aforementioned permissions that always appear in the HTML together are presented to the user with a single message. This will also be discussed in more detail in Section 3.4, but it suggests that Facebook intends for them to always be given together. This further supports that the permissions in the HTML are the ones that Facebook will actually give the site, not just the ones the developer places in the scope.

3.3.2. Differences from what the developer requested Through experimentation with our test site, we determined that several permissions are given in groups. That is, if one permission in a group is put in the scope parameter, all permissions in the group are present in the HTML and are actually granted upon user approval. The permissions that are grouped together are listed in Table 1.

Read Permissions
user_activities, *user_about_me*
friends_activities, *friends_about_me*
email, *contact_email*
read_stream, *export_stream*
Write Permissions
create_note, *upload_photos*, *upload_videos*, *publish_actions*, *publish_checkins*, *publish_stream*, *share_item*, *status_update*
Extended Permissions
rsvp_event, *create_event*

Table 1: Permissions granted in groups.

All of the affected read and extended permissions are grouped in pairs, so if the developer requests one they receive the other. The write permissions, however, are in a group of eight: If a developer asks for one, they get the other seven as well. These are the only eight write permissions offered by the Facebook Connect API—that is, every site that attempts to request one write permission will get them all.

When we crawled the logins of the real sites, we recorded all of the permissions in the HTML in addition to those in the scope (which were requested by developers). Figure 4 shows the number of occurrences in the scope and in the HTML for each of the permissions that are granted in groups. It accounts for all 177 sites crawled with the current login format. It is evident that many sites are being made to request more permissions than the developer specified, especially in the write permissions category. All permissions not listed in the table had the same number of occurrences in the scope and HTML, with the expected exception of *public_profile* and *user_friends* which are given by default.

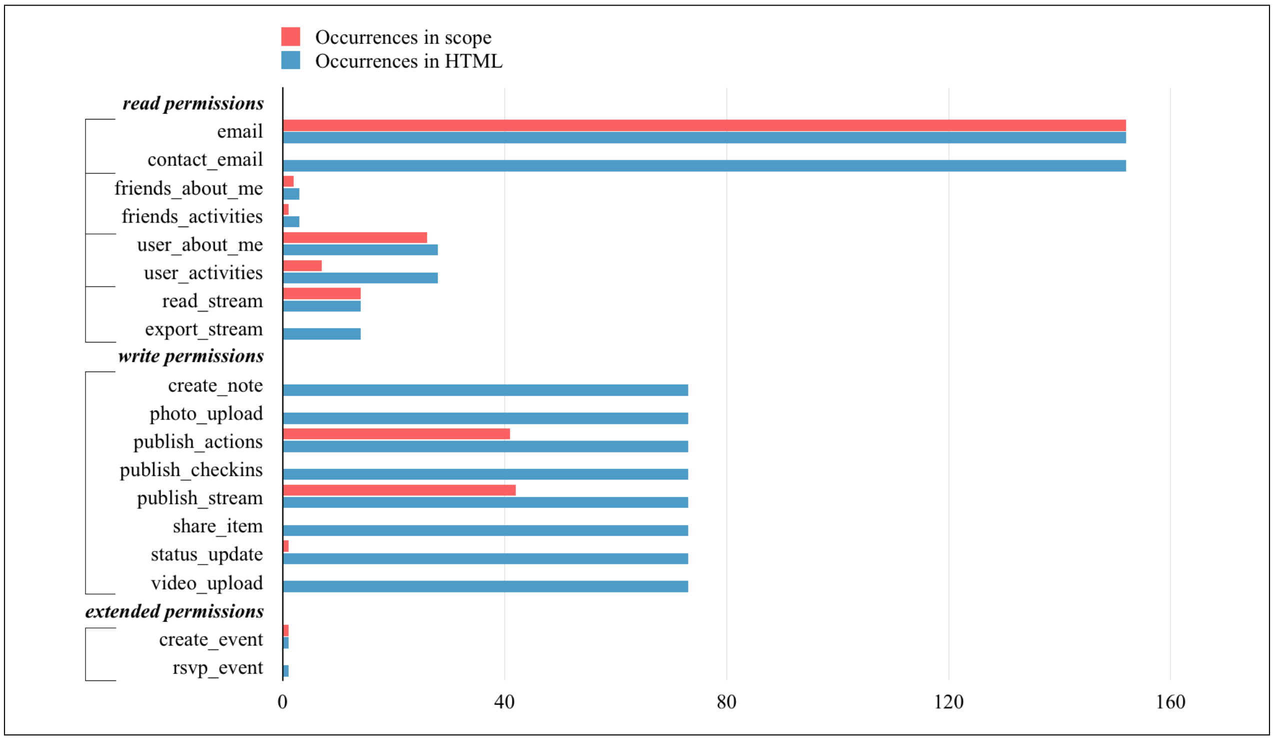


Figure 4: Number of occurrences in the scope and in the HTML for permissions granted in groups. Those in the scope were requested by developers, those in the HTML would actually be given if approved by the user. The lines on the left designate the groups as listed in Table 1.

3.4. How permissions are presented to the user

As mentioned previously, when the user logs in to a site with Facebook Connect they are presented with up to three messages from Facebook asking them to approve the read, write, and/or extended permissions. These messages are algorithmically formed from the permissions present in the corresponding HTML input elements. We identified the phrase or word in the message that corresponds to each permission with our test site and verified the correspondences with the data from the 177 real sites. Most messages appear reasonably clear. However, the permissions that are always presented in groups (see Table 1) have one corresponding word or phrase in the message indicating that *all* the permissions in that group are being requested. Table 2 presents these potentially unclear messages, the corresponding groups of permissions being requested, and the meaning of the permissions according to the Facebook Connect documentation [10]. Similar tables for all permissions can be found in Appendix A.

3.5. Discussion

This section has explored the way a set of permissions is transformed from the time when the developer designs the site to request permissions to the point where the permissions are presented to the user for approval. The developer requests a certain set of permissions and then the Facebook Connect API may add additional permissions to that set because certain permissions are always granted in groups. These are the permissions that will be given to the site should the user approve them. When the user is asked for approval, they are presented with a message that can be broken down into pieces representing each permission or group of permissions.

But does it actually make a difference that some permissions are grouped together in this way? Combining *create_event* and *read_event* and presenting them as "manage your events" seems reasonable, as does combining *email* and *contact_email*, since it is unclear what the difference would be. Although the explanation of *export_stream* is not very clear, it appears reasonable to combine it with *read_stream*. The only obvious issue is that the documentation states that *read_stream* allows access to the News Feed and Wall [10], whereas the message presented to the user says only "News Feed."

Combining *user_about_me* and *user_activities* and calling it "personal description" (as well as the similar combination of *friends_about_me* and *friends_activities*), however, may be confusing. These two do not intuitively go together, and calling them "personal description" hides the presence

Read Permissions: *Site_Name will receive the following info. . .*		
Message	**Permission**	**Meaning** [10]
email address	email	email
	contact_email	not listed
News Feed	read_stream	access my News Feed and Wall
	export_stream	export my posts and make them public. All posts will be exported, including status updates.
personal description	user_about_me	about me
	user_activities	activities
...and your friends'. . .		
personal descriptions	friends_about_me	'about me' details
	friends_activities	activities
Write Permissions: *Site_Name would like to. . .*		
Message	**Permission**	**Meaning**
post to Facebook for you. *– or* –* post publicly to Facebook for you. *– or* –* post privately to Facebook for you.	create_note	create and modify events
	photo_upload	add or modify photos
	publish_actions	publish my app activity to Facebook
	publish_checkins	publish checkins on my behalf
	publish_stream	publish content to my Wall
	share_item	share items on my behalf
	status_update	update my status
	video_upload	add or modify videos
Extended Permissions: *Site_Name would like to. . .*		
Message	**Permission**	**Meaning**
manage your events	create_event	create and modify events
	rsvp_event	RSVP to events

Table 2: Message decoder for permissions that are granted in groups. Decoder tables for all permissions are in Appendix A. Italic text represents how the permissions are introduced when presented to the user. See Figure 1 for an example.
***Which of the three messages is presented depends on to whom the posts will be visible. This is controlled by the menu in the bottom left of the middle image in Figure 1.**

of *user_activities*. Potentially the most confusing is the combination of all eight write permissions into one group that is presented as "post to Facebook for you." Users are likely unaware of the many different things they are allowing the site to do to their profile by clicking okay.

Grouping permissions may also discourage people from using Facebook Connect. The Facebook Connect documentation itself states that developers should "only ask for the permissions that are essential to an app [or site]" [11]. Their research has demonstrated that "the more permissions an app requests, the less likely it is that people will use Facebook to log into [that] app." But since

Facebook Connect often requests more permissions than the developer intended to (even just by always requesting *public_profile* and *user_friends*), the developer has no choice but to request unnecessary permissions.

It appears that the reason that all write permissions are presented together is that Facebook has decided to eliminate the distinction between different types of publishing. The description in the documentation for *publish_actions* is “publish my app activity to Facebook” and the description for *publish_stream* is “publish content to my Wall” [10]. These are quite vague, and seem as though they could encompass nearly anything. A blog post from a Facebook employee [4] helps explain these permissions: They are essentially the same thing (and are being merged into one) and allow a site to do any type of publishing to Facebook. The post mentions that they can be used to upload a photo, which one may have suspected required the *photo_upload* permission. Another post from a Facebook employee mentions that developers should only request *publish_actions* because it encompasses all other write permissions in an effort to “simplify the model” [24]. Furthermore, Facebook’s Graph API lists *publish_actions* as the permission needed for all API calls that involve publishing [9].

This transition towards only one type of publishing is visible to anyone who has used Facebook for several years: updating one’s status and uploading a photo used to be distinct actions, but now they are both performed by creating a post on one’s Timeline. The read and extended permissions that are presented in groups may stem from similar changes in Facebook’s structure and it may no longer be possible to separate them. However, these changes may make the Facebook Connect authentication process less clear to users. Section 4 will explore whether they do.

3.6. Facebook’s response

We sent a security bug report to Facebook stating that we could use the *publish_actions* permission after requesting any other write permission. Facebook Security stated[3] that “this behavior is by design” and confirmed that when one permission is requested in the scope, they “translate them to a broader set of [permissions] which are easier for users to understand” [1]. When asked why

[3]The full correspondence with Facebook Security is in Appendix C.

this was done for write permissions but not read permissions, they responded that they “made this change to simplify the experience for developers and for users” and that “write permissions are more similar...whereas read permissions are more distinct.” We decided to test whether all-or-nothing write permissions are in fact easier for the user to understand.

4. Users’ understanding of permissions

As mentioned in Section 2, Egelman [6] determined that 88% of users have a general understanding of Facebook permissions; however, he studied only the read permissions dialogues. The results of our investigation indicated that confusion is more likely to arise with the write permissions since they are all grouped together and presented with a vague message. We conducted a survey to test this. We hypothesize that users understand Facebook’s read permissions messages better than they understand Facebook’s write permissions messages.[4]

4.1. Methodology

The goal of our survey is to present users with messages that they might see when logging in to a site using Facebook Connect and test whether they understand what permissions they are giving access to by authorizing the login. This can be statistically analyzed by comparing respondents’ success at identifying which permissions they are authorizing with the expected result of random guessing. The null hypotheses that will be tested are as follows:

1. Respondents’ ability to identify which read permissions they are authorizing is no different than if they were randomly guessing.
2. Respondents’ ability to identify which write permissions they are authorizing is no different than if they were randomly guessing.
3. Respondents’ ability to identify which read permissions they are authorizing is no different than their ability to identify which write permissions they are authorizing.

[4]Extended permissions were not tested since they are presented similarly to read permissions and are relatively rare (only seven out of the 177 sites requested extended permissions).

All surveys were conducted on Amazon's Mechanical Turk, a service where workers can be paid to complete simple online tasks. This allowed a large and reasonably diverse response pool for little cost. (See Section 4.3.1 for a discussion of limitations.)

4.1.1. Pilots We piloted three different methods of testing user comprehension. After verifying that the respondent had seen a Facebook Connect login, all pilots began by presenting the respondent with either a read or write permissions message that they might see when using Facebook Connect. No respondents were presented with both to ensure that no one got the two questions mixed up.

Respondents were then presented with one of the following three question types:

1. A yes or no question asking if the site would be able to do something if they clicked okay, such as view their photos or update their status.
2. A list of things the site might be able to do if they clicked okay. The user was asked to select all those they thought the site would be able to do.
3. A free response question asking the user to describe what information they thought the site would be able to see on their profile or what it would be able to do to their profile if they clicked okay.

The free response question would be the ideal way to test user understanding because the other two options prompt the user with ideas that may not have occurred to them otherwise. However, pilots showed that answers to free response questions were frequently too vague to be useful and that respondents may not have put enough thought into their answer. (While this may be representative of how much users pay attention to messages when they log in to sites, it is not useful for this survey.) There was no noticeable difference in responses between the yes or no questions and the multiple-selection questions, so we chose the multiple-selection format to get results about more permissions.

We also experimented with showing the respondent messages from different sites. There was some indication that the site influenced the responses. For example, people were more likely to think photo-oriented sites like Flickr would be able to do photo-oriented things, such as uploading photos. However, since the point of this survey was not to test how the specific site influenced what permissions people thought were being given, the final survey used the site name "Hooli.com." (Hooli is a fake tech company in HBO's *Silicon Valley*.) The description of the site given to users was a description of a real site, *Splashscore.com*. This was one of the sites piloted and we determined it had an appropriately general-sounding description and could conceivably need a wide variety of permissions. The way the site was presented to users can be seen in Appendix B.

4.1.2. Final survey The final survey was taken by 600 Mechanical Turk workers. All were first asked if they had seen a site use Facebook login before—nearly all had. Half of those who had[5] were presented with Facebook's standard write permissions message followed by 13 options of things they might be giving the site permission to do by clicking okay. Eight of the 13 were taken almost directly from the Facebook Connect documentation's permission descriptions [10], so they were all things the site would be able to do (since Facebook gives all write permissions together). The other five were things the site could not do. They were present not to be tested but to eliminate biases due to an aversion to selecting all available options. The 13 options were presented in 4 different orders and can be seen in Appendix B.3.

The other half were presented with read permissions questions. Since read permissions messages vary, we used messages taken from four different real sites with varying numbers of permissions (*Jabong.com*, *Flickr.com*, *Splashscore.com*, and *TripAdvisor.com*). All were renamed "Hooli.com" Each message was followed with eight or nine options for things the site might be able to do. Four or five options were information on a Facebook profile that the site would be able to see. The other four were either things the site could not see or were write or extended permissions.[6] Again, the incorrect answers were only so the respondent did not have to select all options to be correct. The four different questions can be seen in Appendix B.2. There are too many different read permissions to effectively test them all without exhausting the respondents with too many questions, so the ones tested are some of the more common ones.

[5]Respondents who had not seen a Facebook Connect login were not given the rest of the survey to avoid confusion that may have arisen due to only seeing the login dialogues out of context.

[6]It is difficult to determine what the site cannot see since the user's public profile could contain a lot of information if they have relaxed privacy settings, so only very clear-cut things like seeing private messages could be used.

4.1.3. Additional survey for read permissions In one pilot survey of the free response format, a respondent stated that the site would gain access to only a limited number of permissions because their Facebook settings prevented them from accessing the rest. This suggests a lack of understanding of how the read permissions work: A site can access nearly everything that is public with only the *public_profile* permission [14]. By granting the site additional permissions, a user is giving the site permission to access that information regardless of the user's privacy settings. (Using the test site, we confirmed that we could see all user photo albums regardless of their privacy settings with the *user_photos* permission.)

We created a short survey and collected responses from 100 Mechanical Turk workers to see if this confusion was widespread. The survey presented the user with the permission message for *Imgur.com*, which requests the *user_photos* permission. Users were asked to identify which photo albums Imgur would be able to see if they clicked okay. The options were those marked as visible to the public, those marked as visible to friends, and those marked as visible to only them. (The correct answer is all three.) The survey can be seen in Appendix B.4.

This question can also be statistically tested against random guessing. The null hypothesis is that respondents' ability to identify which privacy levels they are authorizing to be viewed is no different than if they were randomly guessing.

4.2. Analysis of survey results

Our survey showed that users are significantly better than random guessing at identifying which read permissions are being requested. However, they are significantly worse at identifying write permissions. The additional survey demonstrated that users may not understand that they are allowing data to be viewed regardless of its privacy settings.

4.2.1. Read permissions Figure 5 illustrates the percentage of people who correctly identified that each permission would be given to the requesting site after they clicked okay.[7] Table 3 lists the numerical percentages as well as the 2-tailed *p*-value from a binomial test comparing the number of

[7]Only the real permissions being requested are presented. The incorrect answers we made up are not.

people who correctly identified a permission as being requested to the expected value with random guessing: half of the total number of people who were presented with that permission.

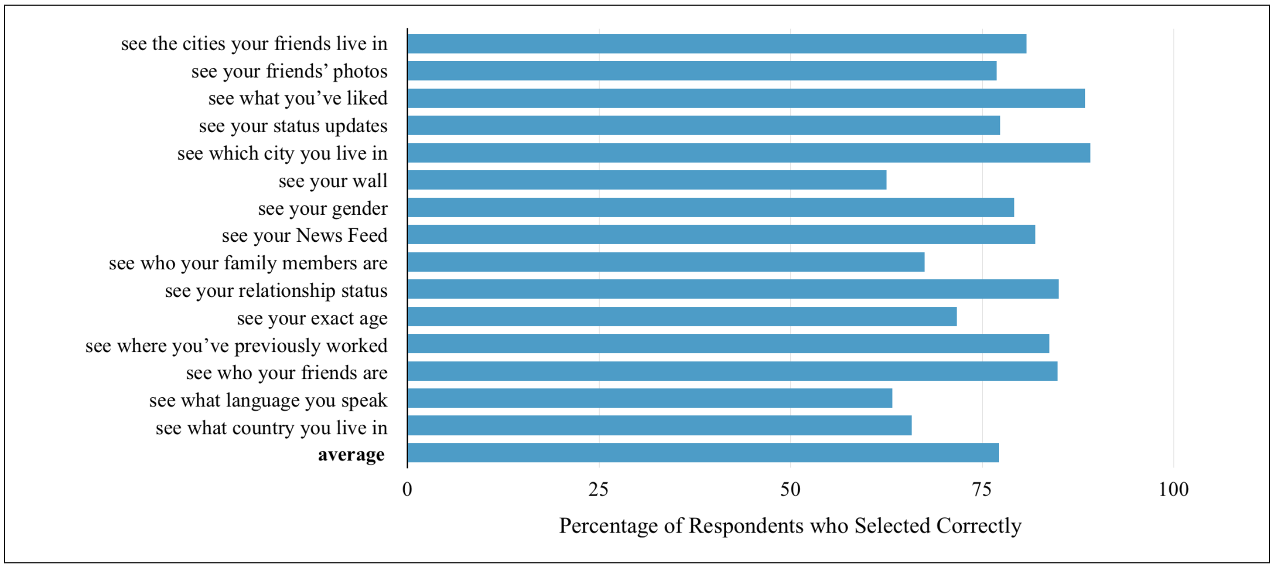


Figure 5: Percentage of people who correctly identified that each permission would be given to the site upon authorization.

Permission	N	Percent Correct	2-tailed *p*-value
see the cities your friends live in	78	80.77	0.000
see your friends' photos	78	76.92	0.000
see what you've liked	78	88.46	0.000
see your status updates	150	77.33	0.000
see which city you live in	230	89.13	0.000
see your wall	72	62.50	0.044
see your gender	72	79.17	0.000
see your News Feed	72	81.94	0.000
see who your family members are	80	67.50	0.002
see your relationship status	80	85.00	0.000
see your exact age	159	71.70	0.000
see where you've previously worked	80	83.75	0.000
see who your friends are	79	84.81	0.000
see what language you speak	79	63.29	0.024
see what country you live in	79	65.82	0.007

Table 3: *p*-values for 2-tailed binomial test comparing the number of people who correctly selected each permission to an expected value of half the total number of people asked.

For all tested read permissions, over half of people correctly identified that said permission would be granted based on the message presented. On average, individual permissions were correctly identified 79.72% of the time. This is comparable to Egelman's [6] conclusion that 88% of users understand generally which permissions are being requested.

The null hypothesis (that respondents' ability to identify which read permissions they are authorizing is no different than if they were randomly guessing) can be rejected for all but two permissions with $p < .01$. This suggests users have a significantly better understanding of which read permissions they are granting than if they were randomly guessing.[8]

The null hypothesis for "see what language you speak" can be rejected with $p < .03$ and for "see your wall" with $p < .05$. A G-test shows that respondents were worse at identifying "see your wall" than "update your status" (which had an accuracy rate roughly equal to the average) with $p < .04$. Recall that seeing one's Wall and seeing one's News Feed are both granted by the *read_stream* permission but the message presented to the user says only "News Feed" (see Section 3.5). This may have been the cause of some confusion. Respondents were also worse at identifying "see what language you speak" with $p < .04$, but the reason for this is unclear.

4.2.2. Write permissions Figure 6 illustrates the percentage of people who correctly identified that each permission would be given to the requesting site after they clicked okay. Table 4 lists the numerical percentages as well as the 2-tailed *p*-value from a binomial test comparing the number of people who correctly identified a permission as being requested to the expected value with random guessing: half of the total number of people who were presented with that permission.

For all permissions except for "publish content to your wall," fewer than half of respondents answered correctly. For all of those except "publish app activity to Facebook," the null hypothesis (that respondents' ability to identify which write permissions they are authorizing is no different than if they were randomly guessing) can be rejected with $p < .01$. That is, for these six permissions, people would have been significantly more likely to correctly identify whether they were granting the permission by randomly guessing.

The *p*-value for "publish app activity to Facebook" is too high to reject the null with a reasonable

[8]It is not the case that users simply marked every survey option as visible to the website: an average of 81.96% of users correctly identified each of the options that would not be visible to the site. The null hypothesis for each of these options can be rejected with $p < .01$.

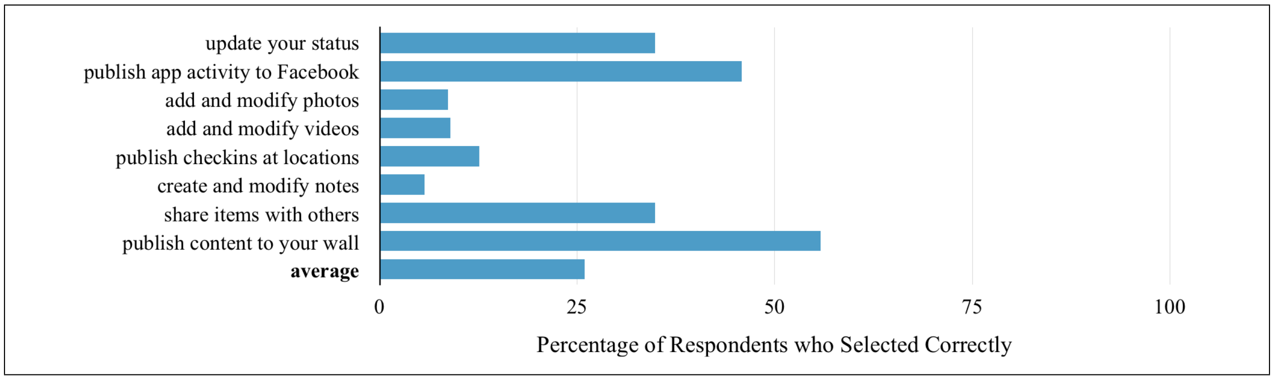


Figure 6: Percentage of people who correctly identified that each permission would be given to the site upon authorization.

Permission	Percent Correct	2-tailed *p*-value
update your status	34.88	0.000
publish app activity to Facebook	45.85	0.166
add and modify photos	8.64	0.000
add and modify videos	8.97	0.000
publish checkins at locations	12.62	0.000
create and modify notes	5.65	0.000
share items with others	34.88	0.000
publish content to your wall	55.81	0.050

Table 4: *p*-values for 2-tailed binomial test comparing the number of people who correctly selected each permission to an expected value of half the total number of people asked. N = 301.

level of confidence.

Over half of people correctly identified that the site would be able to "publish content to [their] wall," and the null can be rejected with $p < .05$. People may have a better idea that this permission is being granted than if they were randomly guessing.

Worth noting is that the two permissions people did best with ("publish content to your wall" and "publish app activity to Facebook") are also the vaguest. (These are the *publish_stream* and *publish_actions* permissions that are intended to give nearly all publishing permissions.) Because they are so vague, the fact the more people selected them correctly probably does not mean that they understand the specific things the site can post on their profile—they include the functions of the other permissions, which most users were not successful at identifying.

4.2.3. Comparison It is evident at this point that users understand read permissions messages significantly better than they understand write permissions messages: Respondents correctly identified whether a read permission would be granted 79.72% of the time, whereas write permissions were only correctly identified 25.91% of the time.

We assigned a ranking to each respondent based on the percentage of permissions they correctly identified[9] and separated them into two groups, one for those asked about read permissions and one for those asked about write permissions. A Mann-Whitney U test of these two groups allows us to reject the null hypothesis that respondents' ability to identify which read permissions they are authorizing is no different than their ability to identify which write permissions they are authorizing with $p < 0.001$.

4.2.4. Additional read permissions survey Our first survey indicates that read permissions are understood decently well. However, the additional survey testing whether people understand that they are giving permission to view things that are not marked as "public" (see Section 4.1.3) suggests that they do not understand this aspect.

Figure 7 illustrates the percentage of people who correctly identified that Imgur.com would be able to see their photo albums with various privacy settings if they clicked okay. Table 5 lists the numerical percentages as well as the 2-tailed p-value from a binomial test comparing the number of people who correctly identified that a privacy level was visible to the expected value with random guessing: half of the total number of people who were given the survey.

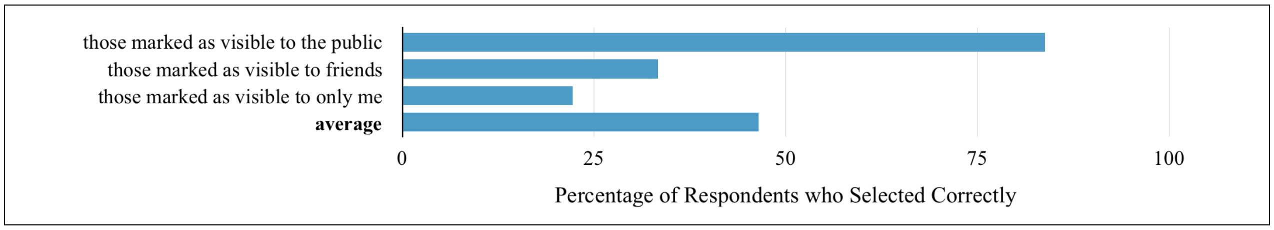


Figure 7: Percentage of people who correctly identified that Imgur.com would be able to see their photo albums of each privacy level upon authorization.

For each privacy setting, the null hypothesis (that respondents' ability to identify which privacy levels they are authorizing to be viewed is no different than if they were randomly guessing) can be rejected with $p < .01$. It appears people are generally aware that they are giving access to their

[9]This counts only the real permissions that will actually be granted and not the incorrect options since we made those up.

Privacy Setting	Percent Correct	2-tailed p-value
those marked as visible to the public	83.84	0.000
those marked as visible to friends	33.33	0.001
those marked as visible to only me	22.22	0.000

Table 5: *p*-values for 2-tailed binomial test comparing the number of people who correctly selected each privacy level to an expected value of half the total number of people asked. N = 99.

photo albums that are marked as public. However, they are generally unaware that they are also giving access to their photo albums that are marked as visible to their friends or only to themselves. This suggests that the relatively high comprehension levels found in my previous survey (79.72%) and Egelman's study (88%) [6] may not actually be entirely representative of user understanding: although people know which types of information they are granting access to, most do not realize they are giving access to that information even if they have marked it with a privacy level other than public. However, this section of our study was performed on a relatively small scale (one website with one permission and 100 responses). A future study could test this comprehensively.

4.3. Discussion

Our survey indicates that users' understanding of write permissions messages is significantly worse than their understanding of read permissions messages. As discussed in Sections 3.5 and 3.6, Facebook claims that all-or-nothing write permissions are easier for the user to understand. However, the comparison to the very granular read permissions suggests that users understand specific, distinct permissions better.

Facebook has attempted to address how general write permissions are by placing responsibility on the developer. The aforementioned Facebook blog post explaining the *publish_stream* and *publish_actions* permissions [4] states that since anything can be shared, "it will continue to be the developer's responsibility to make it clear to the user what content will be shared back to Facebook." It says Facebook's policy was updated to read: "If a user grants [the developer] a publishing permission, actions [the developer takes] on the user's behalf must be expected by the user and consistent with the user's actions within [the] app." As of the time of this writing, this is no longer in the Facebook policy [12].

4.3.1. Limitations There are several possible limitations to this survey:

- As discussed previously, by giving respondents several options to select we suggested possible things the site could do that may not have occurred to them otherwise. In addition, in order to respond to our questions they may have paid more attention to the permissions dialogues than they normally would have. As a result, our survey may indicate that users are more aware of what permissions are being requested than they are in practice.
- If read and write permissions are fundamentally different in some non-obvious way, it may be invalid to compare understanding of read permissions to understanding of write permissions. It is possible that users would understand write permissions even more poorly if they were not all-or-nothing.
- There may be some demographic bias in using Mechanical Turk to collect responses. We did not collect demographic information from respondents although we did restrict respondents to the United States. Our choice of Mechanical Turk is justified by Goodman, Cryder, and Cheema's finding that "[Mechanical Turk] participants produced reliable results that are consistent with previous decision-making research" [16]. The most relevant concern they raise is of respondents not paying enough attention and becoming fatigued in longer surveys; the short length of our surveys hopefully ameliorated that to some degree.
- We had to make up permissions that are not actually granted upon authorization so users did not have to select every option to be correct. If we did a poor job, this could have influenced results. We did not count these made up permissions in our statistical analysis for this reason.

5. Comparison to other SSO implementations

We wanted to compare the way Facebook Connect presents permissions with other SSO implementations. However, the Twitter [23] and Google [17] implementations are not directly comparable. Neither offers near as many permissions as Facebook Connect does. Google has no write permissions at all and Twitter does not separate the read and write permissions so a user has to approve both to log in. In addition, both perform the OAuth authorization on the backend so the permissions being requested are not visible in the HTML as they are with Facebook Connect's JavaScript SDK.

Perhaps the most comparable SSO implementations are older versions of Facebook Connect. However, like Twitter, Facebook Connect previously did not separate the read and write permissions. This was changed in 2013 due to complaints of too many apps posting on user's profiles [5].

The write permissions messages in the older implementations were formatted differently. As mentioned previously, 26 of the 203 websites we crawled used an older implementation. Table 6 presents a sampling of the messages we saw. These messages appear to do a better job indicating what the site can actually post since they provide examples. However, they may be misleading since they provide examples based on the specific site even though all sites in the table request only *publish_actions*. Even though the site may be most likely to post what the message suggests, it can still post anything else.

Site	**Write Permissions Message**
Starpires.com	This app may post on your behalf, including status updates, photos and more.
PioneerLegends.com	This app may post on your behalf, including collections you completed, miles you collected and more.
Stratego.com	This app may post on your behalf, including achievements you earned and more.
OpenShuffle.com	This app may post on your behalf, including your high scores and more.
Fupa.com	This app may post on your behalf, including games you played and more.

Table 6: Write permissions messages from sites using an older version of Facebook Connect. All sites are gaming sites. The only write permission they request is *publish_actions*.

6. Future areas for research

This research could be continued in a variety of directions.

- One could investigate how users' interpretations of the permissions messages are influenced by the site they are logging in to. For example, a user may see the write permissions message on Flickr and think it means the site can upload photos, but someone seeing the same message on Skype may not think Skype can upload photos.

- The results of our survey testing whether users understand that sites are getting access to their information even if it is marked as private (see Figure 7) indicate that there is room for more research in the area. This should be tested with a variety of different permissions. One could also experiment with ways to make it clear to the user that all of their information is being shared, regardless of the privacy settings.
- We previously mentioned that our survey included options of things the sites could not actually do so the respondent would not have to select all options to be correct. However, many people selected these fake options. One could research what permissions users think are being requested beyond what is actually being requested. This is an important area of research because people may be unwilling to use the SSO service if they think too many permissions are being given.
- It is clear that users do not understand the full range of write permissions being requested. However, Egelman [6] determined that users make their decision to use Facebook Connect or not before they see the permissions requested. Egelman only tested read permissions, though. A similar study could see how the presence of write permissions affects users' decisions to use Facebook Connect. It could also test whether explicitly stating what write permissions are being requested and varying the number of write permissions requested affects whether users are willing to use the system.

7. Conclusions

To maximize security and to ensure users feel comfortable using Facebook Connect, developers should be allowed to minimize the number of permissions they request from the user and the permissions should be presented to the user as clearly as possible. On both fronts, Facebook Connect could be improved.

When a developer designs their site to request certain permissions through Facebook Connect, the Facebook Connect system may translate certain permissions into broader groups of permissions that will all be granted if the user authorizes the site to access their profile. This may force users to give unnecessary permissions to a site in order to log in.

The messages presented to the user for read permissions are reasonably clear—our survey showed that a majority of users understand what data they are providing access to. (However, they may be unaware that they are providing access to this information even if their privacy settings are set to private.)

Write permissions, however, are much less clear. Facebook has "simplified" the write permissions process so that every site either gets all write permissions or none. Our survey shows that users do not understand the many things a site will be able to do to their profile if they authorize the vague message stating that the site "would like to post to Facebook for you." Given the relative success with which users were able to identify the more distinct and well-defined read permissions, it appears users might actually understand write permissions better if they were split up.

On April 30, 2014 Facebook announced an update to their Facebook Login system to be rolled out over the following months that allows users to reject individual permissions or log in anonymously [13]. While this is a big step forward, it appears there is still only one publishing permission and it is presented with the same vague message that our survey respondents had trouble understanding. However, it does provide even more specific details about read permissions.

Acknowledgments

Thanks to Joseph Bonneau for his advice and guidance throughout the research process. Thanks to Arvind Narayanan for introducing me to Joe and starting me on this research path. Thanks to Steven Englehardt, Dillon Reisman, Pete Zimmerman, and Christian Eubank for setting me up with the CITP's web crawling infrastructure. (I would also like to acknowledge Steven for discovering the permissions in the hidden HTML input elements.) Thanks to Markus Huber for providing me with the AppInspect dataset [18].

References

[1] Personal correspondence with Facebook Security representative (Neal), April 2014.

[2] "The top 500 sites on the web," Alexa. Available: http://www.alexa.com/topsites

[3] A. Chaabane *et al.*, "A closer look at third-party osn applications: Are they leaking your personal information?" in *Passive and Active Measurement Conference (2014)*. Los Angeles: Springer, March 2014.

[4] L. Chen, *Streamlining publish_stream and publish_actions permissions*, Facebook, April 2012. Available: https://developers.facebook.com/blog/post/2012/04/25/streamlining-publish_stream-and-publish_actions-permissions/

[5] D. Cohen, "Login with facebook update: Apps must now separately request permission to post on behalf of users," August 2013. Available: http://www.insidefacebook.com/2013/08/22/login-with-facebook-update-apps-must-now-separately-request-permission-to-post-on-behalf-of-users/

[6] S. Egelman, "My profile is my password, verify me!: The privacy/convenience tradeoff of facebook connect," in *CHI '13 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM New York, April 2013, pp. 2369–2378.

[7] *Facebook Login*, Facebook. Available: https://developers.facebook.com/docs/facebook-login/

[8] *Getting Started with Custom Stories*, Facebook. Available: https://developers.facebook.com/docs/opengraph/getting-started/

[9] *Graph API Reference*, Facebook. Available: https://developers.facebook.com/docs/graph-api/reference/

[10] *Permissions*, Facebook. Available: https://developers.facebook.com/docs/reference/fql/permissions/

[11] *Permissions with Facebook Login*, Facebook. Available: https://developers.facebook.com/docs/facebook-login/permissions

[12] *Platform Policy*, Facebook. Available: https://developers.facebook.com/policy/

[13] "Introducing anonymous login and an updated facebook login," Facebook, April 2014. Available: http://newsroom.fb.com/news/2014/04/f8-introducing-anonymous-login-and-an-updated-facebook-login/

[14] *Privacy for Apps & Websites*, Facebook, 2014. Available: https://www.facebook.com/help/403786193017893

[15] M. Frank *et al.*, "Mining permission request patterns from android and facebook applications," in *The 12th IEEE International Conference on Data Mining*. Institute of Electrical and Electronics Engineers, December 2012.

[16] J. K. Goodman, C. E. Cryder, and A. Cheema, "Data collection in a flat world: The strengths and weaknesses of mechanical turk samples," *Behavioral Decision Making*, vol. 26, no. 3, pp. 213–224, 2013.

[17] *Using OAuth 2.0 for Login*, Google, March 2014. Available: https://developers.google.com/accounts/docs/OAuth2Login

[18] M. Huber *et al.*, "Appinspect: Large-scale evaluation of social networking apps," in *COSN '13 Proceedings of the first ACM conference on Online social networks*. ACM New York, October 2013, pp. 143–154.

[19] "Report of data protection audit of facebook ireland," Ireland Data Protection Commissioner, December 2011. Available: http://dataprotection.ie/docs/21/12/11_-_Report_of_Data_Protection_Audit_of_Facebook_Irela/1182.htm

[20] P. Sovis, F. Kohlar, and J. Schwenk, "Security analysis of openid," in *Securing Electronic Business Processes - Highlights of the Information Security Solutions Europe 2010 Conference*, 2010.

[21] S.-T. Sun and K. Beznosov, "The devil is in the (implementation) details: An empirical analysis of oauth sso systems," in *Proceedings of ACM Conference on Computer and Communications Security '12*. LERSSE, October 2012.

[22] S.-T. Sun *et al.*, "Investigating user's perspective of web single sign-on: Conceptual gaps, alternative design and acceptance model," *ACM Transactions on Internet Technology*, vol. 13, no. 1, November 2013.

[23] *Sign in with Twitter*, Twitter. Available: https://dev.twitter.com/docs/auth/sign-twitter

[24] A. Wyler, *Providing People Greater Clarity and Control*, Facebook, December 2012. Available: https://developers.facebook.com/blog/post/2012/12/12/providing-people-greater-clarity-and-control/

A. Full message decoding tables

Message	**Permission**	**Meaning** [10]
birthday	user_birthday	birthday
chat status	user_online_presence	online presence
checkins	user_checkins	checkins
current city	user_location	current city
custom friends lists	read_friendlists	access my friend lists
education history	user_education_history	education history
email address	email	email
	contact_email	not listed
events	user_events	events
follows and followers	user_subscriptions	subscribers and subscribees
friend list	user_friends	list of friends
friend requests	read_requests	access my friend requests
groups	user_groups	groups
hometown	user_hometown	hometown
interests	user_interests	interests
likes	user_likes	likes, music, TV, movies, books, quotes
messages	read_mailbox	read messages from my mailbox
News Feed	read_stream	access my News Feed and Wall
	export_stream	export my posts and make them public. All posts will be exported, including status updates.
notes	user_notes	notes
personal description	user_about_me	about me
	user_activities	activities
photos	user_photos	photos uploaded by me
public profile	public_profile	not listed
questions	user_questions	questions
relationship interests	user_relationship_details	significant other and relationship details
relationships	user_relationships	family members and relationship status
religious and political views	user_religion_politics	religious and political views
status updates	user_status	Facebook status
Video activity	user_actions.video	not listed
videos	user_videos	videos uploaded by me
website	user_website	website
work history	user_work_history	work history

Table 7a: Read permission message decoder, part (a). Message begins with *"Site_Name will receive the following info. . . "* See Figure 1 (left image) for an example.

Message	**Permission**	**Meaning** [10]
birthdays	friends_birthday	birthdays
chat statuses	friends_online_presence	online presence
checkins	friends_checkins	checkins
current cities	friends_location	current cities
education histories	friends_education_history	education history
events	friends_events	events
follows and followers	friends_subscriptions	subscribers and subscribees
groups	friends_groups	groups
hometowns	friends_hometown	hometowns
interests	friends_interests	interests
likes	friends_likes	likes, music, TV, movies, books, quotes
notes	friends_notes	notes
personal descriptions	friends_about_me	'about me' details
	friends_activities	activities
photos	friends_photos	photos
questions	friends_questions	questions
relationship interests	friends_relationship_details	significant others and relationship details
relationships	friends_relationships	family members and relationship statuses
religious and political views	friends_religion_politics	religious and political views
status updates	friends_status	Facebook statuses
videos	friends_videos	videos
websites	friends_website	websites
work histories	friends_work_history	work history

Table 7b: Read permission message decoder, part (b). After listing the permissions that apply to the user, the permissions applying to their friends are listed. This part of the message begins with *"...and your friends'. . . "* See Figure 1 (left image) for an example.

Message	**Permission**	**Meaning** [10]
Site_Name would like to post to Facebook for you. *– or –* Site_Name would like to post publicly to Facebook for you. *– or –* Site_Name would like to post privately to Facebook for you.	create_note	create and modify events
	photo_upload	add or modify photos
	publish_actions	publish my app activity to Facebook
	publish_checkins	publish checkins on my behalf
	publish_stream	publish content to my Wall
	share_item	share items on my behalf
	status_update	update my status
	video_upload	add or modify videos

Table 8: Write permission message decoder. See Figure 1 (middle image) for an example. Which of the three messages is presented depends on to whom the posts will be visible. This is controlled by the menu in the bottom left of the middle image in Figure 1.

Message	**Permission**	**Meaning** [10]
access your Facebook ads and related stats	ads_read	access my Facebook ads and related stats
access your Facebook Pages' messages	read_page_mailboxes	read messages for my pages
access your Page and App insights	read_insights	access Insights data for my pages and applications
manage your ads	ads_management	manage advertisements on behalf of me
manage your custom friend lists	manage_friendlists	create, delete, and modify my friend lists
manage your events	create_event	create and modify events
	rsvp_event	RSVP to events
manage your notifications	manage_notifications	may access my notifications and may mark them as read
manage your Pages	manage_pages	manage my pages
send and receive messages on your behalf	xmpp_login	login to Facebook Chat
send you text messages	sms	send SMS messages to my phone

Table 9: Extended permission message decoder. Message begins with *"Site_Name would like to. . . "* See Figure 1 (right image) for an example.

B. Surveys

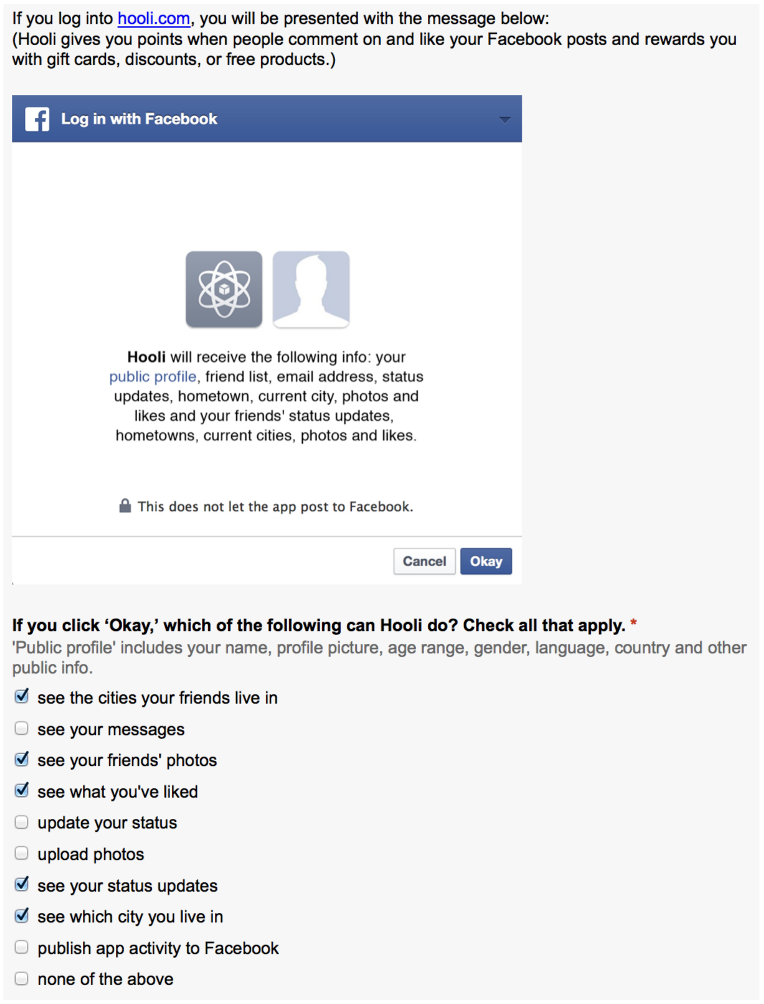
This appendix provides details of the surveys used to test user understanding of permissions messages. Descriptions of the survey process can be found in Section 4.

B.1. Initial question

The first question on every survey reads “Some websites allow you to log in to their site using your Facebook account. Have you seen this?” If the user answered yes, they were taken to the rest of the survey. If they answered no, the survey ended. This was to prevent confusion caused by seeing permissions messages out of context. Nearly all users answered yes.

B.2. Read permissions surveys

Figures 8, 9, 10, and 11 show the four different versions of the survey to test understanding of read permissions. Each uses the fake site name “Hooli” but the permissions are taken from a different real site for each. The correct answers are selected.



If you log into hooli.com, you will be presented with the message below:
(Hooli gives you points when people comment on and like your Facebook posts and rewards you with gift cards, discounts, or free products.)

Log in with Facebook

Hooli will receive the following info: your public profile, friend list, email address, status updates, hometown, current city, photos and likes and your friends' status updates, hometowns, current cities, photos and likes.

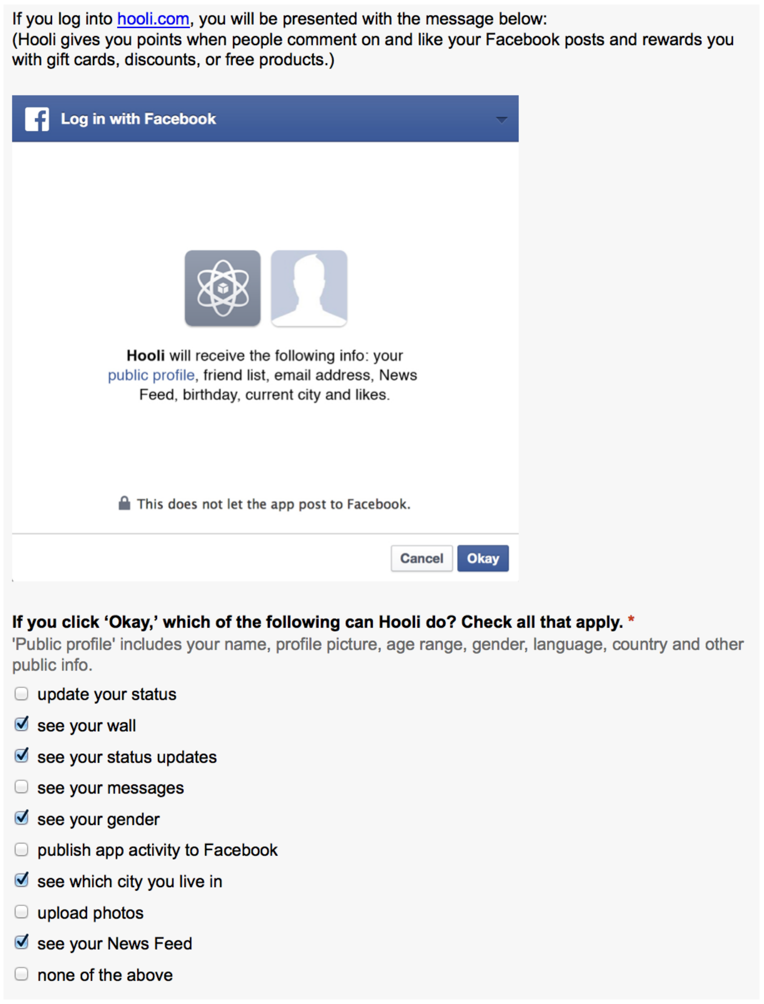
This does not let the app post to Facebook.

Cancel Okay

If you click 'Okay,' which of the following can Hooli do? Check all that apply. *
'Public profile' includes your name, profile picture, age range, gender, language, country and other public info.

- [✓] see the cities your friends live in
- [ ] see your messages
- [✓] see your friends' photos
- [✓] see what you've liked
- [ ] update your status
- [ ] upload photos
- [✓] see your status updates
- [✓] see which city you live in
- [ ] publish app activity to Facebook
- [ ] none of the above

Figure 8: A read permissions survey. The correct answers are selected. This version of the survey uses the permissions from TripAdvisor.com.


If you log into hooli.com, you will be presented with the message below:
(Hooli gives you points when people comment on and like your Facebook posts and rewards you with gift cards, discounts, or free products.)

Hooli will receive the following info: your public profile, friend list, email address, News Feed, birthday, current city and likes.

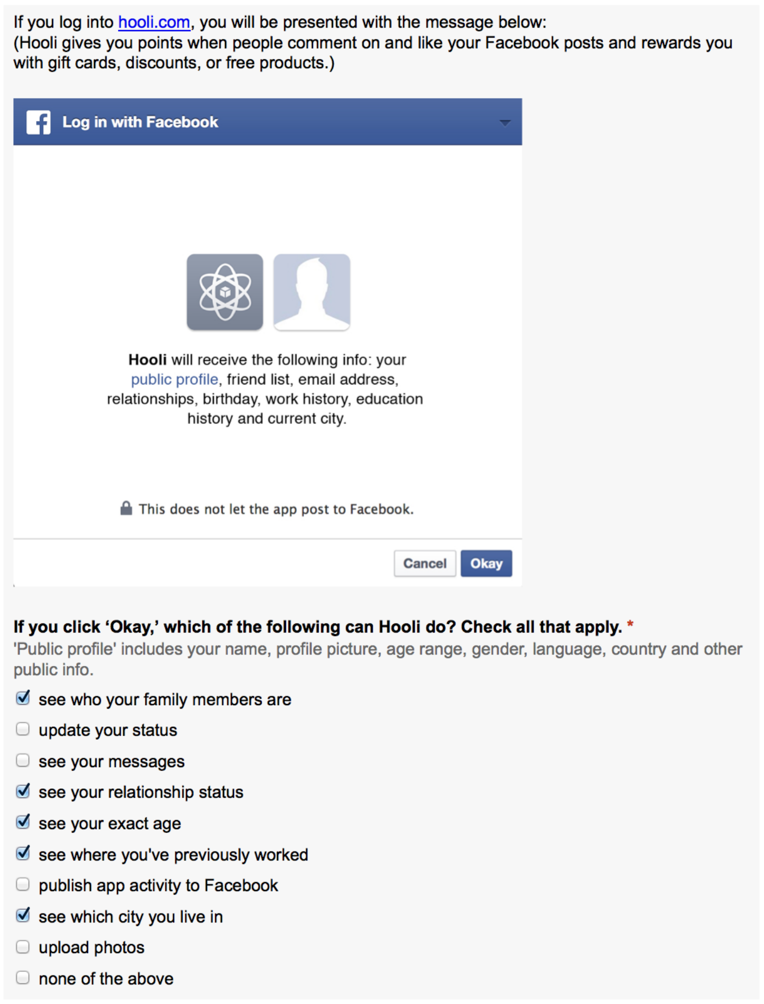
This does not let the app post to Facebook.

If you click 'Okay,' which of the following can Hooli do? Check all that apply. *

'Public profile' includes your name, profile picture, age range, gender, language, country and other public info.

- ☐ update your status
- ✓ see your wall
- ✓ see your status updates
- ☐ see your messages
- ✓ see your gender
- ☐ publish app activity to Facebook
- ✓ see which city you live in
- ☐ upload photos
- ✓ see your News Feed
- ☐ none of the above

Figure 9: A read permissions survey. The correct answers are selected. This version of the survey uses the permissions from Splashscore.com.



If you log into hooli.com, you will be presented with the message below:
(Hooli gives you points when people comment on and like your Facebook posts and rewards you with gift cards, discounts, or free products.)

Hooli will receive the following info: your public profile, friend list, email address, relationships, birthday, work history, education history and current city.

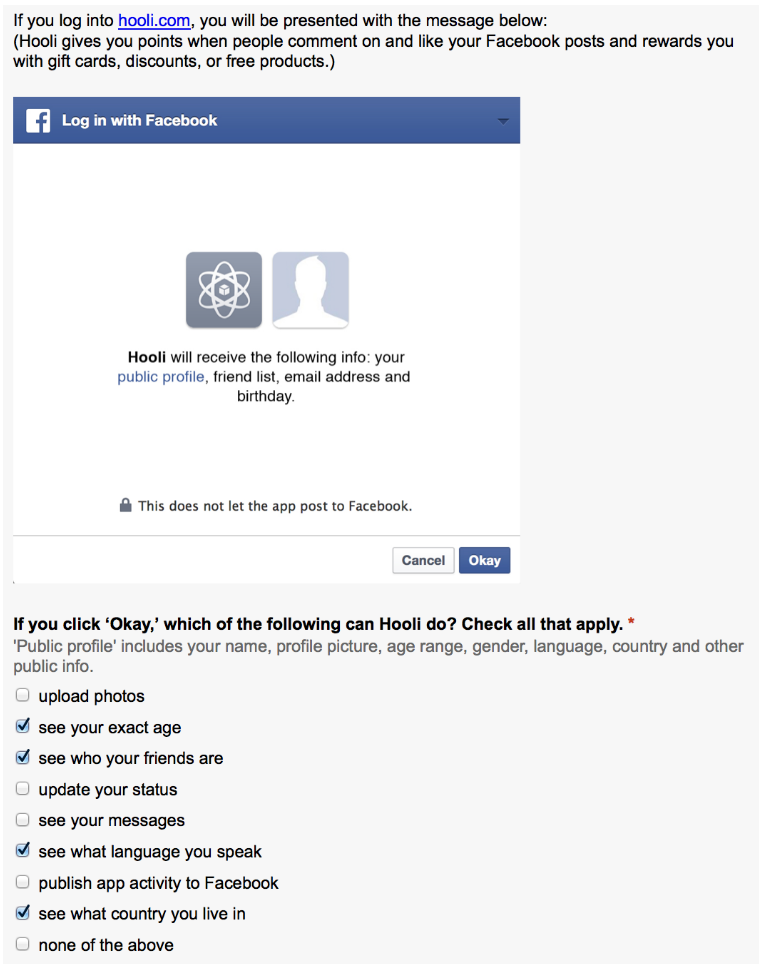
This does not let the app post to Facebook.

If you click 'Okay,' which of the following can Hooli do? Check all that apply. *

'Public profile' includes your name, profile picture, age range, gender, language, country and other public info.

- [✓] see who your family members are
- [ ] update your status
- [ ] see your messages
- [✓] see your relationship status
- [✓] see your exact age
- [✓] see where you've previously worked
- [ ] publish app activity to Facebook
- [✓] see which city you live in
- [ ] upload photos
- [ ] none of the above

Figure 10: A read permissions survey. The correct answers are selected. This version of the survey uses the permissions from Jabong.com.



If you log into hooli.com, you will be presented with the message below:
(Hooli gives you points when people comment on and like your Facebook posts and rewards you with gift cards, discounts, or free products.)

Log in with Facebook

Hooli will receive the following info: your public profile, friend list, email address and birthday.

This does not let the app post to Facebook.

Cancel Okay

If you click 'Okay,' which of the following can Hooli do? Check all that apply. *

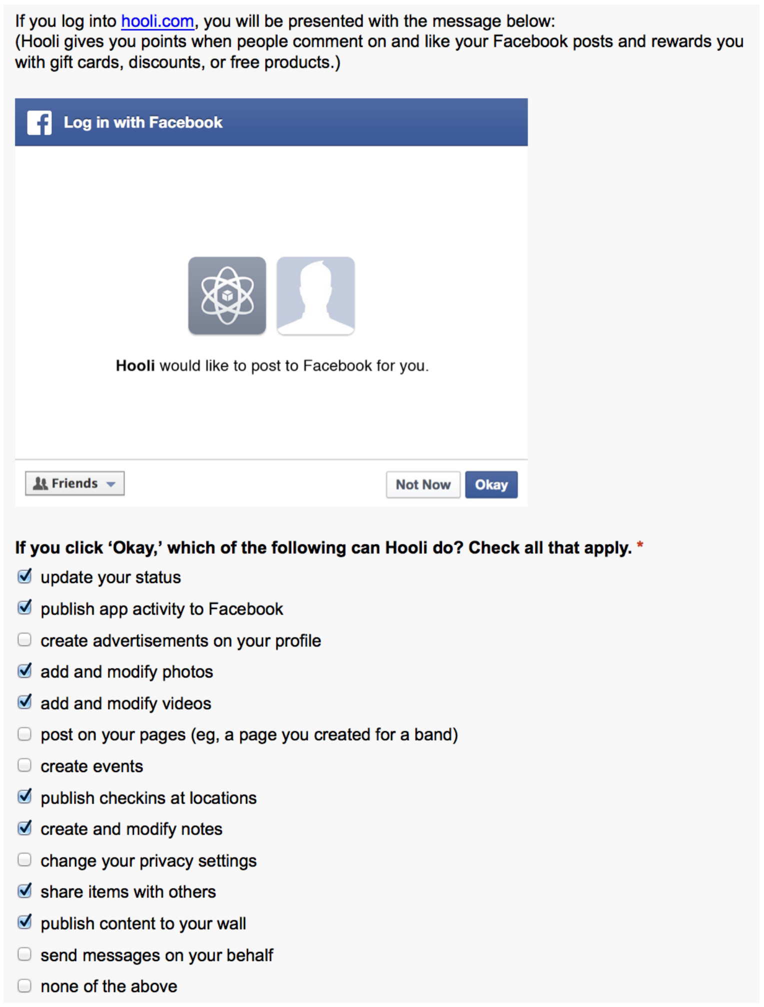
'Public profile' includes your name, profile picture, age range, gender, language, country and other public info.

- [ ] upload photos
- [x] see your exact age
- [x] see who your friends are
- [ ] update your status
- [ ] see your messages
- [x] see what language you speak
- [ ] publish app activity to Facebook
- [x] see what country you live in
- [ ] none of the above

Figure 11: A read permissions survey. The correct answers are selected. This version of the survey uses the permissions from Flickr.com.

B.3. Write permissions surveys

Figure 12 shows the survey to test understanding of write permissions. It also uses the fake site "Hooli." The correct answers are selected. There were a total of four versions of this survey with the options in different orders.



If you log into hooli.com, you will be presented with the message below:
(Hooli gives you points when people comment on and like your Facebook posts and rewards you with gift cards, discounts, or free products.)

Hooli would like to post to Facebook for you.

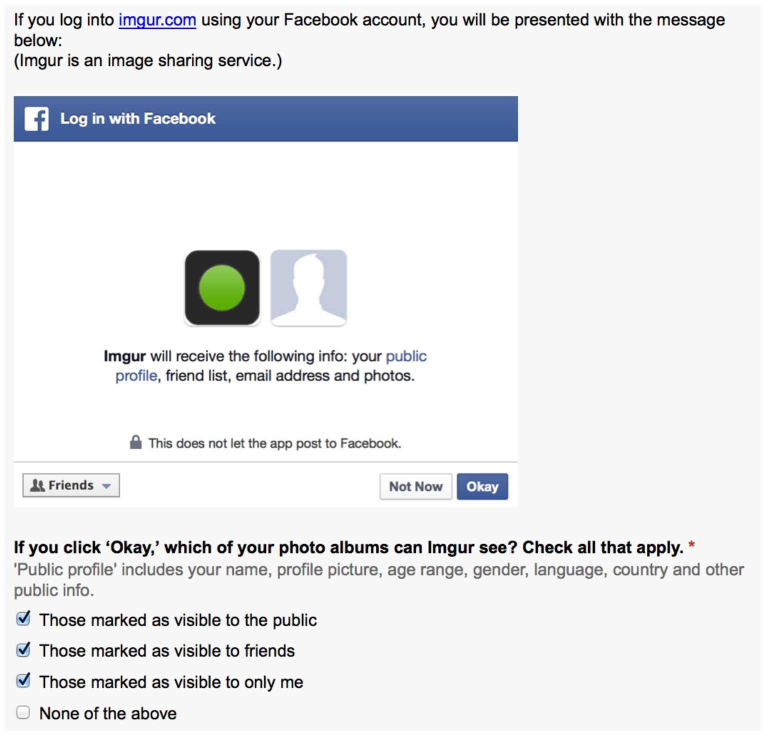
If you click 'Okay,' which of the following can Hooli do? Check all that apply. *

- [✓] update your status
- [✓] publish app activity to Facebook
- [ ] create advertisements on your profile
- [✓] add and modify photos
- [✓] add and modify videos
- [ ] post on your pages (eg, a page you created for a band)
- [ ] create events
- [✓] publish checkins at locations
- [✓] create and modify notes
- [ ] change your privacy settings
- [✓] share items with others
- [✓] publish content to your wall
- [ ] send messages on your behalf
- [ ] none of the above

Figure 12: A write permissions survey. The correct answers are selected.

B.4. Additional read permissions survey

Figure 13 shows the survey to test whether users understand that they are giving access to their information that is not marked as visible to the public. The correct answers are selected.


If you log into imgur.com using your Facebook account, you will be presented with the message below:
(Imgur is an image sharing service.)

Imgur will receive the following info: your public profile, friend list, email address and photos.

This does not let the app post to Facebook.

If you click 'Okay,' which of your photo albums can Imgur see? Check all that apply. *

'Public profile' includes your name, profile picture, age range, gender, language, country and other public info.

- [✓] Those marked as visible to the public
- [✓] Those marked as visible to friends
- [✓] Those marked as visible to only me
- [ ] None of the above

Figure 13: The additional read permissions survey. The correct answers are selected.

C. Correspondence with Facebook Security

As mentioned in Section 3.6, we sent a security bug report to Facebook reporting that we could use the *publish_actions* permission after requesting any other write permission (see Section 3.3.2). Below is the full correspondence with Facebook Security [1].

Initial bug report

Description and Impact:

I can design a site with Facebook Connect that publishes a story with the 'publish_actions' permission. However, if I request any other write/publishing permission, such as 'create_note', I can still use the 'publish_actions' permission and publish the story. I believe this is a vulnerability because applications may be receiving more capability than they believe they are requesting.

Reproduction Instructions / Proof of Concept:

1. I followed the Facebook documentation instructions to create a story with the publish_actions permission: https://developers.facebook.com/docs/opengraph/getting-started/

2. If I replace publish_actions in data-scope with any other write permission, including create_note, I can still publish the story. (If I replace it with a read permission such as email I cannot.)

Facebook Security's response

Thanks for writing in. Can you send in some screenshots of the dialog you see when requesting the different permissions? I'm curious to see if the wording changes between the two.

Our response

Below are screenshots of the two messages presented whether I request create_note or publish_actions. *[screenshots not shown here, roughly equivalent to Figure 1, center image]*

The HTML for these messages has three hidden input elements named read, write, and extended. The permissions requested appear in their value fields. However, if I request any of the 8 write permissions (publish_actions, publish_stream, status_update, video_upload, photo_upload, share_item, create_note, or publish_checkins), all 8 appear in the value of the input element named write. I've been researching this for a class project at Princeton University and I've confirmed that this is true on 73 of 73 different websites that request write permissions. The only two write permissions messages between the 73 sites are "App_Name would like to post to Facebook for you" and "App_Name would like to post publicly to Facebook for you." The presence of "publicly" is just determined by the selection on the menu on the bottom left of the message page (second screenshot), not by the permissions being requested.

Facebook Security's response

I'll confirm with the Platform team, but I believe this is intentional behavior: as you noted, while in the URL you're requesting one scope we actually translate them to a broader set of scopes which are easier for users to understand.

Facebook Security's followup

I just confirmed with our Platform team that this behavior is by design.

Our response

Ok, thanks for looking into that. Is there a reason you do that for the write permissions but not for read or extended permissions?

Facebook Security's response

The Platform team made this change to simplify the experience for developers and for users. My guess would be that generally, write permissions are more similar (ie: creating a note versus creating a video versus posting all are ways to create content on the site that aren't very different) whereas read permissions are more distinct (ie: an app which can view your friends does not necessarily need to view your relationships unless major functionality changes).

[End of correspondence]