Is this App Safe? A Large Scale Study on Application Permissions and Risk Signals

Pern Hui Chia
Q2S NTNU[*]
Trondheim Norway
chia@q2s.ntnu.no

Yusuke Yamamoto
Kyoto University
Kyoto Japan
yamamoto@dl.kuis.kyoto-u.ac.jp

N. Asokan
Nokia Research Center
Helsinki Finland
n.asokan@nokia.com

ABSTRACT

Third-party applications (apps) drive the attractiveness of web and mobile application platforms. Many of these platforms adopt a decentralized control strategy, relying on explicit user consent for granting permissions that the apps request. Users have to rely primarily on community ratings as the signals to identify the potentially harmful and inappropriate apps even though community ratings typically reflect opinions about perceived functionality or performance rather than about risks. With the arrival of HTML5 web apps, such user-consent permission systems will become more widespread. We study the effectiveness of user-consent permission systems through a large scale data collection of Facebook apps, Chrome extensions and Android apps.

Our analysis confirms that the current forms of community ratings used in app markets today are not reliable indicators of privacy risks of an app. We find some evidence indicating attempts to mislead or entice users into granting permissions: free applications and applications with mature content request more permissions than is typical; "look-alike" applications which have names similar to popular applications also request more permissions than is typical. We also find that across all three platforms popular applications request more permissions than average.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access Controls*; H.5.m [**Information interfaces and presentation (e.g., HCI)**]: Miscellaneous

General Terms

Security, Human Factors, Measurement

Keywords

Android Apps, Chrome Extensions, Facebook Apps, Application Permissions, Privacy

[*]Centre for Quantifiable Quality of Service in Communication Systems (Q2S), Centre of Excellence, appointed by the Research Council of Norway, is funded by the Research Council, Norwegian University of Science and Technology (NTNU) and UNINETT.

This version: camera ready (26 Feb 2012). To appear in ACM WWW2012.

1. INTRODUCTION

Ever since the personal computer changed the lives of people around the world, we have become accustomed to the notion of software applications. The personal computer world started out with completely open platforms where all applications (apps) ran with the same complete set of privileges available to the user. This quickly gave rise to the phenomenon of malicious and inappropriate software [10].

Operating system and runtime platform security schemes can be used to apply the principle of least authority to applications. Although various platform security schemes were developed since the 1960s, they saw widespread deployment only when they were incorporated into Java Security Architecture and into mobile device platforms [21]. All modern mobile device application platforms incorporate permission-based platform security schemes. Web application platforms like Facebook and browser extension runtimes like Google Chrome extensions also use permission-based platform security. Some of these platforms such as Apple's iOS rely on a central authority to decide what permissions can be granted to a given application. Others rely on the user making the authorization decisions. We will call the former category *centralized permission systems* and the latter *user-consent permission systems*.

In the smartphone arena, centralized permission systems are currently dominant, with the exception of the Android platform. However several HTML5 APIs (e.g., the geolocation API) support a user-consent permission system. The availability of comprehensive APIs including offline caching makes it possible for HTML5 web apps to offer similar functionality as native applications. If HTML5 web apps become dominant [23], their decentralized nature will also imply that user-consent permission systems will become more widespread.

Centralized permission systems take the burden of judgment away from users. While this is a benefit in terms of usability, it is problematic if the judgment in question is subjective. People and organizations may disagree on whether a certain application is privacy-invasive or offensive [8]. The biggest problem in user-consent permission systems is that users may become habituated to permission queries and may carelessly click through them. Even careful users have to make access control decisions based only on a few signals such as the average numerical rating given by other users, number of ratings and downloads. Users who care about their privacy [7, 1], may not have the ability to protect it in user-consent permission systems if the signals are unreliable or can be manipulated by developers of malicious apps.

In this paper, we investigate the current state of risk signaling on privacy intrusiveness of apps, and if there is any evidence of attempts to mislead or entice users of user-consent permission systems into compromising their privacy. Specifically we ask:

1. Do popular apps ask for more permissions than is typical for apps in general?
2. Are currently available signals about an application reliable in indicating privacy risks associated with that application?
3. Do developers of free apps and those with mature content ask for more permissions than is typical?
4. Do apps with "look-alike names" (i.e., names similar to popular apps) ask for more permissions than is typical?

Our paper is structured as follows. We start with a survey of related work in Section 2 and describing the data collection processes in Section 3. We present a basic analysis on app popularity, ratings and permissions in Section 4 before proceeding to evaluate the effectiveness of current risk signals (Section 5) and potential trends of enticements and tricks (Section 6). We conclude by revisiting the above four questions and discussing the implications and mitigation measures in Section 7.

2. RELATED WORK

With the growing popularity of Android, there have been a number of publications on Android OS security and its permission system. Enck et al. [12] proposed Kirin certification to help identifying Android apps that request a suspicious permission combination using a set of predefined rules. Barrera et al. [6] studied the relationship between the permissions requested by 1,100 most popular and free Android apps by machine learning and proposed a methodology to improve the expressiveness of app permissions without increasing its overall complexity. Our study differs from theirs in that we do not look into the patterns of permission requests in details but we study the how the number of permissions requested by apps correlate with several signals (e.g., community ratings) that the users receive.

Felt et al. [17] studied the effectiveness of permission systems on Android and Chrome platforms. Using 1,000 most popular Chrome extensions, they pointed out that the first 500 most popular extensions have requested significantly more permissions than the second 500 extensions. However, they observed no differences between the 756 most popular and 100 most recent Android apps. With a much larger dataset, our study shows that there is a positive correlation between the number of installations and the number of permissions requested by the app on all three web and mobile application platforms: Facebook, Chrome and Android. In addition, we look into whether specific types of apps, including those with mature content, those flagged by external ratings and those with suspicious look-alike names, request for more permissions than is typical.

In comparison, fewer studies have investigated the Facebook permissions. King et al. [20] conducted a survey on the privacy knowledge, behaviors and concerns of Facebook app users. Our study differs from theirs in two ways. First our analysis relies on a large scale data collection rather than a self-reported survey. Second while they focused on the interaction between user's understanding, concerns and behaviors when using Facebook apps, we look at the availability of reliable risk signals on Facebook (as well as Chrome and Android) and how the absence of them may have been exploited by some developers to entice or trick the users with questionable apps.

Moore and Edelman [24] studied the ecosystem of the typo-squatting fraud – the intentional registration of misspellings of popular website addresses. They estimated that at least 938,000 typo domains targeted the 3,264 popular .com sites they studied. They also found that 80% of the typo-squatting domains were supported through profits from advertising, typically from the pay-per-click advertisements. Our study is one of the first to analyze the naming exploitations in apps. While we have not conducted an in-depth analysis on the motivation and profitability of such naming tricks, we analyzed whether apps with look-alike names request for more permissions than is typical. A related work is by Barrera et al. [5] which studied the problem of a non-global app ID system for Android apps. They proposed *Stratus* to standardize the app IDs across different Android marketplaces. Our analysis in this paper focuses on look-alike names rather than look-alike app IDs. We expect the users to pay less attention to app IDs especially on Facebook and Chrome where the app IDs are in the form of a string of random digits or characters.

3. DATA COLLECTION

We detail the data collection processes for Android apps, Chrome extensions and Facebook apps in the following. We share our datasets on our project site [25].

3.1 Android Apps

Prior studies on Android OS security (e.g., [6, 12, 16, 17]) have mainly focused on the most popular apps on Android Market. In order to broaden the scope we used both the official Android Market [3] and AppBrain.com [4] to construct two datasets `Android (pop)` and `Android (new)`.

`Android (pop)` consists of popular Android apps selected randomly from the *top-selling-free* and *top-selling-paid* listings of Android Market, as well as the list of the most popular apps according to AppBrain.com on 15 June 2011. After removing duplicates, it contains 650 unique apps (323 paid and 327 free). `Android (new)` consists of 1210 new apps (610 paid and 600 free) which first appeared on the *most-recent-apps* section of AppBrain.com in mid June 2011 and were still available in early October. We kept track of these new apps and updated our database accordingly for changes in app details.

In addition to the above, to investigate the behavior of apps with look-alike names, we collected also a separate much larger dataset `Android (mr)`. The dataset consists of 20,500 new Android apps (11,095 free and 9,405 paid) constructed from the list of 20 most recent apps according to AppBrain.com on an hourly basis from mid August to early October 2011.

The application information page on Android Market provides a number of details that we use in this paper, including app installation count, average community rating, rating count, developer URL, price, content maturity level, and permissions requested by the app.

3.2 Facebook Apps

We constructed the Facebook dataset by downloading the entire list of 34,370 Facebook apps (app names, IDs, and developer IDs) from SocialBakers [26], a portal providing the usage statistics of various social media. We then attempted to access the Facebook application page of each of these (using the Watir [28] library to login to Facebook). We excluded apps that have become unavailable or otherwise invalid or redirected to a page outside Facebook. Out of the remaining 27,029 Facebook apps, 18,205 request at least one permission from the user. For each of the 27,029 apps, we downloaded details including the number of monthly active users, the average rating, rating count, description and category. This constitutes the `Facebook (all)` dataset.

3.3 Chrome Extensions

Chrome Web Store [19] lists up to 1,000 most popular extensions in 12 different categories. As some of the categories such as 'Sports' and 'Shopping' have far less than 1,000 extensions, even the more recent extensions such as those with less than 10 users, were present on the lists. We constructed our dataset by downloading all 12 lists. Removing duplicates and extensions that became unavailable during data collection, the resulting `Chrome (all)` dataset consists of 5,943 extensions. It contains details from the information page of each extension, including the installation count, average community rating, rating count, developer URL, version of extension, supported languages, as well as the permission warnings associated with the extension.

4. BASIC ANALYSIS

We first study the link between app popularity and user ratings, and the statistics of permissions in the following.

4.1 App Popularity and User Ratings

We define the *popularity* of an app as the number of installations (in the case of Android apps and Chrome extensions) or the number of monthly active users (in the case of Facebook apps). Figure 1 shows log-log plots of app popularity versus the number of user ratings the app has received. While we did not test if both the distributions of app popularity and the number of ratings per app follow a specific heavy-tailed distribution (e.g., Power-Law, Log-normal), they appear to be highly skewed visually. All four app popularity distributions curve in the log-log plot. On the other hand, other than the `Android (pop)` dataset where no apps have less than 30 ratings, the rating contribution patterns appear to be straight lines in the log-log plot, suggesting that they could be a Power Law. Indeed, many online peer production systems can be characterized by a Power Law contribution pattern that is explainable by the participation momentum rule [31]. The skewed nature of the app popularity and the number of ratings per app prompted us to use the logarithmic values, i.e., $log(\#installation)$ and $log(\#rating)$ for popularity and rating count respectively in subsequent analysis.

We found a strong (Pearson) correlation between app popularity and the number of ratings of the app has received (with r ranging from 0.67 to 0.90, $p<.001$, see Figure 2). Indeed, users are more likely to rate an app that they have installed. However, somewhat counter-intuitively, the average rating of an app, $avgr$, does not positively correlate with app popularity. In fact, $avgr$ is negatively and weakly correlated to the app popularity ($r{=}-0.15$, $p<.001$) among the new Android apps. We figure that the average user rating can indeed be misleading without factoring in its confidence level. Considering the confidence of an average rating to be proportional to the number of ratings that it has received, we thus measure the adjusted average rating to be:

$$avgr_a = (avgr - 3) * log(\#rating) \qquad (1)$$

The –3 transformation is necessary as user ratings range from 1 to 5 across the different app platforms. As also shown on Figure 2, there is a strong correlation between the app popularity and the adjusted average rating (with r ranging from 0.45 to 0.72, and $p<.001$). We evaluate if $avgr_a$ serves as an effective risk signal against potentially intrusive or inappropriate apps in Section 5.

4.2 Permission Statistics

Understanding the complex permission models of web and mobile apps can be a daunting task for many ordinary users. We briefly describe the permission systems from the most granular (Android) to the least (Chrome) below. In each case, we identify three sets: the set of **all permissions** P, the set of **dangerous permissions** P_{danger}, and the set of **dangerous and information relevant permissions** P_{dinfo}. P_{danger} consists of permissions for actions that can be potentially harmful to the user while P_{dinfo} is the subset of P_{danger} consisting of permissions that permit access to sensitive personal information of the user.

Android: Android permissions are categorized into 4 categories, namely *Signature-or-System*, *Signature*, *Normal* and *Dangerous*. The first two categories, *Signature-or-System* and *Signature* permissions protect the most sensitive operations on the Android devices. These permissions can only be granted to apps pre-installed into the device's `/system/root` folder and/or apps signed with the device manufacturer's private key. Requests to use these permissions by other apps without the right keys will be ignored [17]. On the other hand, the *Normal* permissions govern the functionalities which can be annoying (e.g., vibrating the phone), while the *Dangerous* permissions protect the user from operations that can be potentially harmful including those that cost money or potentially privacy intrusive [17]. The details of individual Android permissions can be found on [2].

We observed 137 different permissions in our dataset, out of which 65 are dangerous permissions. Following [15], we classified 34 of the 65 permissions in P_{danger} as belonging to P_{dinfo}. Table 1 shows the most frequently requested dangerous permissions among the popular and new Android apps. The full list is available on our project site [25].

Facebook: The Facebook permissions have evolved as the platform and its user base grows. Each of the permissions protects a specific piece of personal information or a specific functionality on the platform (e.g., to login to Facebook chat system, publish user's activity on his wall). We consider all Facebook permissions to be dangerous. There are in total 62 Facebook permissions [13], which are grouped and presented to the users in 23 different categories. Each category is highlighted in bold and visualized with a unique icon on the permission request screen, while the individual permissions are described in gray text with a smaller font size under their respective category. The individual permissions requested can also be found on the URL of the per-

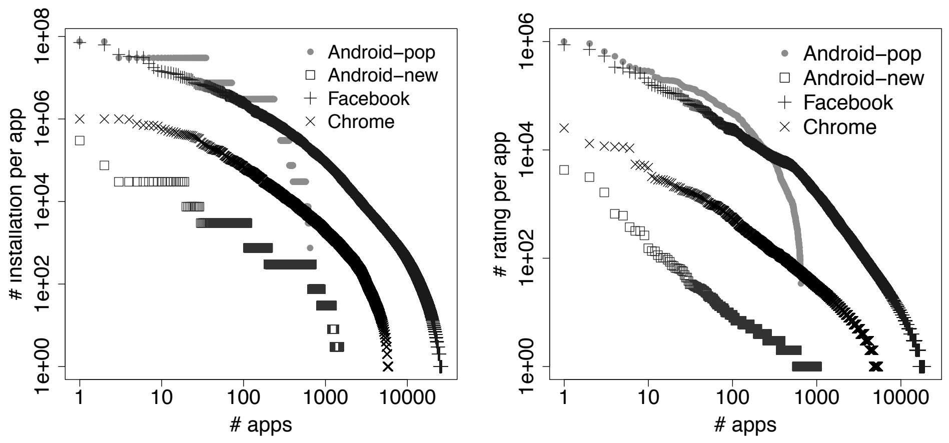

Figure 1: Distribution of the number of installations and number of ratings per app.

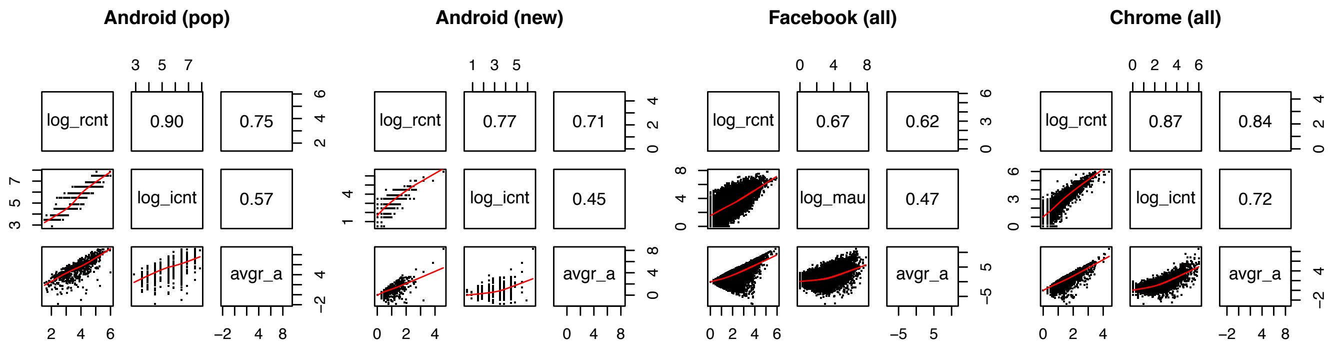

Figure 2: Scatter-plot matrices showing the pairwise correlation coefficients (upper right triangular panel) and the scatter-plots (lower left) of any given pairs of three variables: log of installation count, `log_icnt` (or monthly active users, `log_mau`), log of rating count, `log_rcnt`, and adjusted average rating, `avgr_a`. The scales of the variables are depicted on the borders of individual matrices. All correlation values are statistically significant with p<.001.

	pop	new
INTERNET	77	67
WRITE_EXTERNAL_STORAGE†	51	34
READ_PHONE_STATE†	45	29
WAKE_LOCK (5)	23	13
ACCESS_FINE_LOCATION† (4)	14	14
ACCESS_COARSE_LOCATION†	10	5
READ_CONTACTS† (8)	7	5
CAMERA† (11)	6	3
READ_LOGS† (17)	6	1
RECORD_AUDIO† (20)	6	1
GET_TASKS† (10)	6	4
CALL_PHONE (7)	5	5

Table 1: Top 12 most requested dangerous permissions w.r.t. `Android (pop)`, and percentage of apps requesting them. Numbers in brackets show the different rankings w.r.t. `Android (new)`.

	%
Access my basic info.†	67
Post to Facebook as me.	23
Send me email.†	14
Access my profile info.†	5
Access my data any time.†	2
Access my photos.†	2
Access information people share with me.†	2
Publish games and app activity.	2
Access posts in my news feed.†	1
Access my videos.†	1

Table 2: Top 10 most requested permissions, and percentage of Facebook apps requesting them.

† indicates a dangerous and info. relevant permission

	%
Your tabs & browsing activity.	58
Your data on <*some*> websites.†	41
Your data on all websites.†	35
Your bookmarks.†	2
Your browsing history.†	2
Your list of installed apps, extensions, and themes.†	1
Your physical location.†	1
All data on your computer and the websites you visit.†	1

Table 3: Percentage of Chrome extensions requesting a specific permission.

Perm. type	Android (pop)				Android (new)				Facebook (all)				Chrome (all)			
	μ	M	max	total	μ	M	max	total	μ	M	max	total	μ	M	max	total
P	4.5	4	15	137	3.0	2	20	137	1.2	1	13	23	1.4	2	6	8
P_{danger}	3.0	3	10	65	2.1	2	11	65	1.2	1	13	23	0.8	1	5	7
P_{dinfo}	1.6	1	7	34	1.0	1	5	34	0.9	1	11	14	0.8	1	5	7
R_p (%)				91				74				67				85

Table 4: Mean (μ), median (M) and maximum (max) number of P, P_{danger} and P_{dinfo} requested per app, as well as the total number of permissions on different platforms. R_p measures the percentage of apps requesting at least one permission.

mission request screen but we do not expect the ordinary users to notice them. For these reasons, in this study, we have focused on the permission categories as opposed to the individual permissions. For simplicity, we refer to Facebook *permission categories* as *permissions* interchangeably.

14 out of the 23 permission categories are information relevant. Table 2 shows the most frequently requested permission categories. Notice all apps requesting a permission must also request for the *'Access my basic information'* permission. The most dangerous permission is perhaps *'Access my data any time'* which allows an app to access the user's information even when he is not online. Another interesting permission is *'Access information people share with me'* which allows the app to access *not* information about the user himself, but the personal information of his friends.

Chrome: Among the three platforms, Chrome permissions are the least granular. There are only 9 permission warnings in total as detailed on [18]. Inline with the categorization of Android and Facebook permissions, we regard all Chrome permissions to be both *dangerous* and *dangerous-and-information-relevant*, except the permission *'Your tabs and browsing activity'* which was unnecessarily required for creating a new tab in earlier versions of Chrome browser [18].

The most dangerous is the permission to access *'All data on your computer and the websites you visit'* which is requested by the plugin-type extensions. These plugin extensions are basically native executables that run with full privileges on the user's machine. They are manually reviewed by Chrome before being accepted to appear on the Chrome Web Store [18]. We found only 47 plugin extensions in our dataset. Among the extensions that request for the permission to access *'Your data on <some> websites'*, the top 10 most frequently requested sites are: `google.com`, `facebook.com`, `tiny-url.info`, `plus.google.com`, `twitter.com`, `youtube.com`, `mail.google.com`, `g2me.cn`, `api.flickr.com`, and `reddit.com`.

Summary: Table 4 shows the mean, median and maximum number of P, P_{danger}, and P_{dinfo} that an app requests in the different datasets. The ratio of apps requesting at least a permission is high across all four datasets: `Android (pop)` (91%), `Android (new)` (74%), `Facebook (all)` (67%) and `Chrome (all)` (85%). Another trend that holds across the different platforms is that most apps request only a small fraction of the total available permissions. On average (medium), Facebook and Android apps request for only $1/23 = 4\%$ and $3/65 = 5\%$ (3% among the new apps) of the total available dangerous permissions respectively. In addition, there is a noticeable trend that some permissions are more frequently requested than the others (see Table 1, 2, 3). These reinforce the findings by Felt et al. [17] that an application permission system has the benefits of allowing the platform owners (i) to avoid granting the full privileges to third party apps, and (ii) to possibly recognize apps with anomalous permission request patterns for triaging the manual review process.

	Correlation with $log(\#installation)$			
Perm. type	Android (pop)	Android (new)	FB (all)	Chrome (all)
P	0.26	0.12	0.28	0.15
P_{danger}	0.26	0.11	0.28	0.12
P_{dinfo}	0.22	0.10	0.28	0.12

Table 5: Correlation between app popularity and the number of P, P_{danger} and P_{dinfo} requested. All values are statistically significant with p<.001

However, as it is with other security problems, the permission systems will not be effective if users do not comprehend how they work, or if the permission systems contradict other signals the users receive. One can easily exploit the lack of understanding and the absence of reliable risk signals for questionable or malicious purposes.

5. EFFECTIVENESS OF RISK SIGNALS

We look into the availability of reliable and intuitive risk signals during the process of app installation in the following.

5.1 App Popularity

Table 5 shows that there is a weak positive correlation between app popularity and the number of permissions requested. The trend holds true not only for our Chrome extension dataset (which is much larger than the dataset used by Felt et al. [17]), but also for Android and Facebook apps. Also, the correlation is stronger in popular Android apps than in new Android apps. As Felt et al. [17] hypothesized, a possible explanation is that popular apps need more permissions in order to offer more functionality that makes them more interesting or useful and hence popular. While this phenomenon is perhaps not surprising, it underlines the fact that careful users concerned about their privacy have to make a tradeoff between the functionality offered by an app and its potential for compromising their privacy. Although popularity of an app is an easily available and understandable signal to users of app marketplaces, it is not necessarily a reliable signal for privacy risk – a user cannot conclude that a popular app is a safe one.

5.2 Community Rating

Community rating reflects how the users perceive an app. As with the popularity measure, it is a meaningful signal that is also widely available in app marketplaces. If it is

	Correlation with $avgr_a$			
Perm. type	Android (pop)	Android (new)	FB (all)	Chrome (all)
P	0.15	0.08*	0.11	0.18
P_{danger}	0.14	0.06°	0.11	0.16
P_{dinfo}	0.11	0.04°	0.12	0.16

Table 6: Correlation between the adjusted average rating $avgr_a$ and the number of P, P_{danger} and P_{dinfo} requested. All values are statistically significant with p<.001, except for the case of new Android apps where * indicates p<0.1 and ° indicates p>0.1.

to be an effective signal for enabling the user to detect privacy risks, it would exhibit a negative correlation with the number of permissions requested. We found no such negative correlation between the adjusted average rating and number of permissions (Table 6). In fact there was a weak positive correlation in all cases except for the case of the new Android apps, where the correlation was not statistically significant. Again, the likely explanation is that user ratings in app marketplaces are based on functional aspects like features and performance rather than privacy risks.

5.3 External Ratings

Next, we studied how ratings from sources external to the marketplaces relate to the number of permissions. We considered two sources. First is the Web of Trust (WOT) [29] service. WOT is a community rating system which allows users to rate a website in four dimensions: trustworthiness, vendor reliability, privacy and child safety. It aggregates user ratings as well as information from other sources into a rating along each of the dimensions as well as a combined score. WOT ratings are usually given at the granularity of fully qualified domain names, unless a subdomain has received enough input ratings on its own. Websites where multiple users control their own subsections thus typically share a common rating inherited from the parent domain. Second is AppBrain.com which is a website for discovering Android apps. It provides a number of useful listings including the most popular Android apps in different countries and age groups, as well as the latest apps that appear in Android Market. To help its users, AppBrain.com labels apps created by developers who have made a high fraction of apps without any rating or with below average ratings, as *spam*.

We studied how WOT rating of the website of the app developer (where available) relates to the permissions requested by it. Table 7 shows the average number of permissions requested by apps whose developer websites are deemed suspect (*bad* or *caution*) by WOT. We classified a WOT rating into *bad*, *caution*, *good* or *unknown* following the default risk signaling strategy of WOT as detailed in [9]. We ignored the *good* ratings from WOT as many developers use a shared domain as their website and WOT's verdicts will not be accurate in these cases. Contrasting with the mean values in Table 4, we see that that the suspect apps consistently request more permissions than on average in all cases. The differences in the average number of P, P_{danger} and P_{dinfo} between suspect Facebook apps and Facebook apps in general are statistically significant with p<.001. The sample size is too small in the case of Android apps (see Table 7). As for Chrome extensions, the differences are statistical significant with p<.05 for P, while p<.1 for P_{danger} and P_{dinfo}. Further, we found no correlation between the WOT's suspect rating and community rating.

Perm. type	Android (pop)	Android (new)	FB (all)	Chrome (all)
P	6.0	3.0	3.4	1.8
P_{danger}	3.6	2.2	3.4	1.0
P_{dinfo}	2.0	1.2	2.6	1.0
# apps	5	11	88	65

Table 7: Average number of P, P_{danger} and P_{dinfo} requested by apps whose developer website has been labelled as *bad* or *caution* by WOT.

New Android apps which have been regarded as spam by AppBrain.com also request for a higher number of permissions on average with P, P_{danger} and P_{dinfo} equal 3.3, 2.2, and 1.2 respectively (as compared to 3.0, 2.1 and 1.0 typically, as shown in Table 4). The differences in the average number of P and P_{dinfo} are significant with p<.1.

5.4 Signals from the Developer

So far we have looked at aggregated signals available in the marketplace (popularity and community rating) and signals from external sources. Now we turn our attention to signals that originate from the developers themselves. We considered three different signals:

Availability of a developer website: The availability of a developer website of Android apps and Chrome extensions correlates positively with the number of permissions requested by an app. Thus, the presence of a developer website (developer identity) does not imply a less intrusive app; in fact the reverse was observed. We have not measured the same effect for Facebook apps as the developer website is not shown on the user-consent permission dialogs. One may be able to obtain the developer website in the *Contact Developer* link on the app information page. However, we found that, in many cases, the link provides a means to contact the developer via Facebook's messaging system, rather than a valid developer website.

Availability of a privacy policy: The availability of a privacy policy with a Facebook app correlates negatively, albeit weakly, with P_{danger} (r=–0.12, p<.001) and P_{dinfo} (r=–0.14, p<.001). In other words, there is some weak evidence that Facebook apps accompanied by a privacy policy are more likely to request fewer permissions. Note that the privacy policy URLs were obtained from the user-consent permission dialogs. We have not looked for the privacy policy URLs of Android apps and Chrome extensions as they are not readily available.

Multiple apps from the same developer: Surprisingly, the number of apps a developer has published is negatively correlated to both $log(\#installation)$ and $avgr_a$ among Facebook apps, Chrome extensions and new Android apps. The more apps a developer publishes, the more likely his apps have a lower popularity and community rating. This could be due to that prolific developers actually make low quality apps, or that users actually cast a higher expectation on regular developers; it may be worth further investigation. The number of apps a developer makes has no correlation with the number of permissions their apps request except for the case of new Android apps, where there is a very

	Corr. with *maturity*		Corr. with *price=free*	
Perm. type	Android (pop)	Android (new)	Android (pop)	Android (new)
P	0.30	0.27	0.22	0.43
P_{danger}	0.33	0.30	0.23	0.41
P_{dinfo}	0.30	0.32	0.19	0.35

Table 8: Correlation between the number of P, P_{danger} and P_{dinfo} requested by Android apps, the content maturity level and whether an app is free. All values are statistically significant with p<.001.

Perm. type	Android (pop)		Android (new)	
	paid	free	paid	free
P	3.9	5.2	1.6	4.1
P_{danger}	2.6	3.5	1.1	2.8
P_{dinfo}	1.4	1.9	0.5	1.4
R_p (%)	86	96	56	92

Table 9: Average number of P, P_{danger} and P_{dinfo} requested by the free and paid Android apps. R_p measures the percentage of apps requesting at least one permission.

weak link (r=–0.09, p<.01) between the number of apps the developer has made and the number of permissions. Thus, one cannot judge the potential privacy intrusiveness of an developer based on the number of apps he has published.

6. ENTICEMENTS AND TRICKS

We investigated if there is any evidence of attempts to entice or mislead the user into granting sensitive permissions. In Section 6.1 we study if free apps or those containing mature content require more privileges than average. In Section 6.2 we study the permission request patterns of apps whose names look similar to the popular ones.

6.1 Free and Mature Apps

Android Market requires the developer of an app to rate its content maturity by selecting one of four labels describing the age of the target audience: everyone, low, medium or high maturity. Table 8 shows that there is a positive correlation between the content maturity rating and the number of permissions required. There is also a positive correlation between the requested permissions and whether the app is free. Table 9 shows the difference between paid and free apps in terms of the average number of permissions they request: free apps consistently request more permissions than paid apps. Note that also there is a bigger proportion of free apps requiring at least one permission than the paid apps; this is particularly evident among the new Android apps. Previous studies [6, 17] found that more free apps request for the `INTERNET` permission, possibly only to load advertisements. It is interesting to note that the `INTERNET` permission is not part of P_{dinfo} in our analysis. The consistently higher number of P_{dinfo} of free apps among both popular and new Android apps suggests some suspicious enticement. We further compared between free and paid apps excluding permissions that are commonly required due to third party advertisement libraries. Not counting `ACCESS_COARSE_LOCATION`, `ACCESS_FINE_LOCATION`, `ACCESS_NETWORK_STATE`, `READ_PHONE_STATE`, `WAKE_LOCK` as well

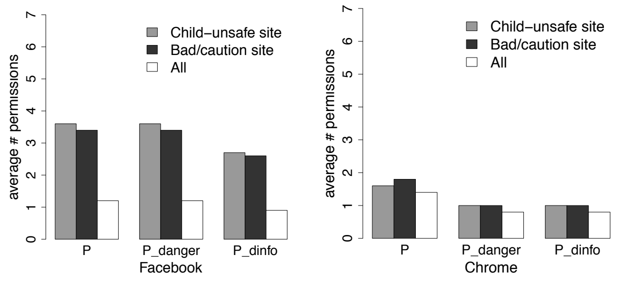

Figure 3: Average number of P, P_{danger} and P_{dinfo} requested by Chrome extensions and Facebook apps whose developer site has been identified as *bad/caution* or *child-unsafe* by WOT.

as `INTERNET`, we found that free apps still request for a higher number of P, P_{danger} and P_{dinfo} on average. The differences are statistical significant with p<.01 in all cases (except for P_{dinfo} of popular Android apps where p<.05).

Facebook apps and Chrome extensions are always free. There is also no content rating systems for these. We divided these apps into three sets in terms of the WOT ratings of the developer: those labeled as child-unsafe, those labeled as suspect (*bad* or *caution*) (as we have discussed in Section 5.3), as well as the set of all apps. The first category (child-unsafe) consisted of 34 chrome extensions and 70 Facebook apps. The second category (suspect sites) consisted of 65 chrome extensions and 88 Facebook apps (also shown in Table 7). Figure 3 shows the results. Suspect apps and apps with potentially child-unsafe content request more permissions than is typical. This effect is particularly pronounced in the case of Facebook apps where all differences are significant with p<.001. Meanwhile, the differences in the average number of P_{danger} and P_{dinfo} between the set of Chrome extensions whose developer website has been identified as child-unsafe by WOT and the set of all extensions are significant with p<0.1.

6.2 Look-Alike App Names

Apps are uniquely identifiable on the respective application platforms through unique strings, such as `102452128776` for FarmVille on Facebook, `com.rovio.angrybirds` for Angry Birds on Android, and `bbncpldmanoknoahidbgmkgob-gmhnafh` for Last.fm extension on Chrome. However, unique identifiers are typically long and unintuitive. One would thus expect the users to recall or discover an app through its name or other visually distinctive features, rather than the IDs. On the application platforms, developers are free to choose his preferred app name; the app names need not be unique even on individual platforms. This creates opportunities for exploitation, for example, to create "look-alike" apps with names exactly the same or similar to the popular ones, so to free ride on their success or as a means to distribute potentially malicious apps.

We looked into the problem of name look-alike apps on Android, Chrome and Facebook. To measure name similarity, we have used the popular Damerau-Levenshtein edit distance [11, 22]. Given two strings s_1 and s_2, we define the normalized edit distance as follows:

$$dist_n(s_1, s_2) = \frac{dist_{DL}(s_1, s_2)}{max(len(s_1), len(s_2))} \qquad (2)$$

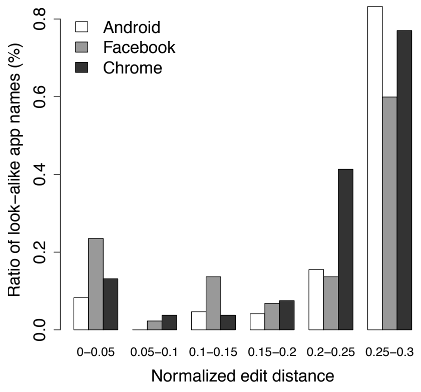

Figure 4: Ratio of look-alike app names in specific ranges of normalized edit distance, $dist_n$

where $dist_{DL}(s_1, s_2)$ is the Damerau-Levenshtein distance between s_1 and s_2, $len(s)$ is the length of string s in terms of the number of characters, and $max(a, b)$ returns the larger value between a and b.

We outlined the top 200 most popular apps on Facebook, Chrome and Android respectively, and calculated the normalized name edit distance between them and the rest of the apps. We ignored the name pairs where both apps are published by the same developers. We also omitted apps with non-Latin based names in this study. Together, we have investigated 19,344 Android apps, 13,181 Facebook apps, and 5,322 Chrome extensions.

Figure 4 shows the ratios of apps whose normalized edit distance to any of the popular counterparts falls in a specific range. As shown by the first three bars (i.e., $dist_n \leq 0.05$), there is a higher ratio of extremely look-alike apps on Facebook than on Android and Chrome platforms. Using a similarity threshold of $dist_n$=0.30, we found 1.15% of the Android apps, 1.20% of the Facebook apps, and 1.47% of the Chrome extensions have either intentionally or unintentionally used a look-alike name to a popular app.

Next, we manually categorized the look-alike app names into the following five classes:

- **Same:** Exact same name of the original counterpart
- **Letter change:** Some letters of a term of the original counterpart is changed
- **Term change:** Some terms of the original counterpart is substituted with new terms
- **Term addition/deletion:** Some terms are added into or deleted from the original counterpart
- **Serialization:** Special terms indicating a different version (e.g., *2*, *Free*) are added to the original

Table 10 lists some examples of look-alike names we have found, while Table 11 gives the percentage breakdown of the different classes of look-alike names. We found a total of 223, 158 and 78 look-alike names on Android, Facebook and Chrome respectively. Going through the list manually, we observed that the look-alike names in the *Same*, *Letter change* and *Serialization* classes are created with a higher level of questionable intention. We delved into look-alike apps of these three classes in the following.

	Popular App	Look-alike App
Letter change:		
A	Tap_Fish	TapFish
A	Chess Free	WChess free
F	FarmVille	SarmVille
F	PhotoMania	Pho.to Mania
C	Facebook Notifications	Facebook Notification
Term change:		
A	Advanced Task Killer	Advanced Task Manager
A	Blue Skies Live Wallpaper	Blue Wave Live Wallpaper
F	Angry birds	Angry bears
F	FarmTown	FameTown
C	Google +1 Button	Google Plus Button
Term addition/deletion:		
A	Yahoo! Mail	My Yahoo! Mail
A	Ringtone Maker	MP3 Ringtone Maker
F	Yearbook	myYearbook
C	Facebook Notifications	Facebook Chat Notification
C	Reader Plus	Reader to Plus
Serialization:		
A	Advanced Task Killer	Advanced Task Killer Pro
F	Pool Master 2	Pool Master
F	Daily Horoscope	Free Daily Horoscopes
C	Reader Plus	ReaderPlus+
C	Speed Dial 2	Speed Dial 2 (ru)

Table 10: Example look-alike app names of different classes on Android (*A*), Facebook (*F*) and Chrome (*C*). Pairs of exactly the same names are not shown.

Similarity class	Android	Facebook	Chrome
Same	6.7	19.6	7.7
Letter change	2.7	23.4	11.5
Term change	78.9	47.5	65.4
Term addition/deletion	4.0	5.7	9.0
Serialization	7.6	3.8	6.4

Table 11: Percentage breakdown (%) of look-alike names in different similarity classes.

Table 12 compares the characteristics of suspect look-alike apps (*Same*, *Letter change* and *Serialization*) to the set of targeted original counterparts, and the set of all apps in general. As shown on the first three rows, the average number of P, P_{danger} and P_{dinfo} requested by the look-alike apps are in general lower than the original counterparts. However, the differences are not statistically significant (except for P_{dinfo} of Facebook). While this suggests that the look-alike apps are not more privacy-intrusive than the original counterparts (i.e., the popular apps), we cannot rule out the potential risks of these look-alike apps immediately. Indeed, the average number of permissions requested by the look-alike apps are higher that is typical (i.e., comparing with the set of all apps). The increase in the average number of permissions is statistically significant for both Android and Facebook look-alike apps. This suggests some level of suspicious activities among the look-alike apps on Android and Facebook platforms.

We further analyzed how community ratings respond to look-alike apps currently. First, we found that the adjusted average rating, $avgr_a$ of the targeted counterparts is significantly higher than that of the look-alike apps across three platforms. However, we should not assume that community ratings are warning against look-alike apps adequately. The

Perm.	Android			Chrome			Facebook		
type	look-alike	original	all	look-alike	original	all	look-alike	original	all
P	3.9	4.2	3.0***	1.8	2.3	1.4	2.1	2.5	1.2***
P_{danger}	2.6	2.7	2.1**	1.2	1.6	0.8	2.1	2.5	1.2***
P_{dinfo}	1.3	1.5	1.0**	1.2	1.6	0.8	1.5	1.9*	0.9***
$avgr_a$	2.2	7.4***	2.3	1.7	3.3***	1.2	1.8	4.2***	1.1***
$avgr$	3.8	4.5	4.4	4.1	4.2	4.4	3.9	4.0	3.8

Table 12: Comparing the look-alike apps to (i) the original counterparts, and (ii) all apps as a whole. Rows 1–3 compare the average number of P, P_{danger} and P_{dinfo} requested. Rows 4–5 compare the adjusted average rating $avgr_a$, and the average rating $avgr$. ***$p<.01$, **$p<.05$, and *$p<.1$ indicate if a measurement given by the original counterparts, and the set of all apps, is significantly different from that of given by the look-alikes.

higher $avgr_a$ value of the targeted apps can be likely due to the higher number of user ratings the apps have received, following their popularity. Indeed, we found no significant differences between the average rating, $avgr$ of the look-alike apps and the targeted counterparts. Also, the average rating of the look-alike apps are not low, ranging from 3.8 to 4.1. In addition, the adjusted average rating of the look-alike Facebook apps is significantly higher than that of all Facebook apps in general. These suggest the lack of the current community rating systems in signaling against the look-alike apps, especially on Facebook. Facebook does not even present the number of user ratings nor app popularity on the user-consent permission dialogs.

7. DISCUSSION AND CONCLUSIONS

Revisiting the questions we started with in Section 1, we summarize and discuss the implications of our findings, and provide recommendations in the following:

1. Popular apps request more permissions: Dissecting the API calls of 940 apps, Felt et al. [16] found that one third of them request for *unused* permissions, attributable to errors and confusion over the insufficient API documentation. Without a source- or binary-code analysis, we have not singled out the cases where apps request for more permissions due to developer errors rather than questionable intentions. Yet, does not matter the causes (errors or bad intentions), unfortunately, there appears to be no disincentives for developers who over privilege their apps currently. There is in fact a positive correlation between app popularity and the number of permissions the app requests on all three platforms, even when considering information sensitive permissions only. More worrying is that the trend holds true despite the different UI designs and permission granularities of Facebook, Chrome and Android. Ongoing research in improving risk communication (e.g., [27]) must take into account the high permission request frequency by popular apps to be effective.

2. No reliable app risk signals currently: As users are 'trained' to accept the requests from popular apps, permission systems can become more ineffective over time. The problem is compounded by the fact that the currently available signals about an app are unreliable in indicating the privacy risks associated with that app. We investigated several such signals including the adjusted community rating, the availability of a developer website and the number of apps published by the developer. If they are to be reliable signals for helping users detect privacy intrusive apps, they should exhibit negative correlation with the number of dangerous permissions requested (and hence with potential privacy intrusiveness of the app). None of the above signals exhibit the expected negative correlation. The only exception we found is the presence of a privacy policy on the permission request screen of Facebook apps that weakly correlates with a lower number of requested permissions. However, if users start relying on this as a signal, it could lead to adverse selection as malicious developers can easily put up a 'privacy policy' that they do not adhere to.

On the other hand, we found some external services that show potential in signaling app risks. One is the website reputation scores from the Web of Trust (WOT) and another is the flagging of spam Android apps by AppBrain.com. App marketplaces can prominently display signals from similar sources to help users recognize potentially intrusive apps. Facebook is already receiving the website reputation scores from WOT to protect against malicious URLs posted onto the users' wall [14]. It will not be too difficult to adapt the scores to warn against suspicious apps and developers.

3. Enticement of free and mature apps: We found evidence indicating attempts to mislead or entice users into granting permissions with free apps and mature content. The trend holds even when focusing on information relevant permissions only. Particularly, excluding the `INTERNET` permission necessary for advertisement revenue (and a few others commonly required by third party ad libraries), free apps still request for more permissions than the paid apps.

4. Look-alike name trick: We also found "look-alike" apps to request more permissions than is typical. While the fraction of look-alike apps is small, there is an underlying problem of 'cheap identity' with app names (and IDs) currently. Charging for a developer ID or for publishing an app may help, but platform owners may be reluctant to do so in the competitive market to attract developers and apps.

An option is to leverage on community inputs for reputation scores on app security and privacy. WhatApp.org [30] is a website which collates user and expert reviews on the privacy, security and openness of web and mobile apps. However, it is still in its beta version and has not attracted much reviews to date. Indeed there are many challenges in crowdsourcing of security and privacy inputs. As we found in this paper, the number of ratings is highly correlated to the popularity of an app. This gives rise to the question of who will review suspicious apps or grayware? There is probably no one size that fits all. We see that a successful model will need to combine community inputs with automated evaluations.

Limitations and future work: One limitation of our analysis is that we have not done any source- or binary-code analysis of the apps. While we pointed out the trends that free, mature and look-alike apps request more permissions than is typical, we cannot directly infer the maliciousness of a particular app judging from the permissions it requests. Secondly, while our analysis confirms the higher risk with free and mature apps, and that there is a lack of reliable signals, we are not sure if users (in particular mature app users) are actually aware of the privacy risks and making the tradeoff willingly. Studies to examine the privacy tradeoff of users will be interesting. Leveraging on large datasets, we also plan to explore the use of machine learning methods for automatic classification of app privacy intrusiveness.

8. ACKNOWLEDGMENTS

This work has benefited from initial discussions with Adrienne Porter Felt and David Barrera. We are also grateful to Adrienne and the anonymous reviewers for their valuable comments on earlier drafts.

9. REFERENCES

[1] A. Acquisti and R. Gross. Imagined communities: Awareness, information sharing, and privacy on the facebook. In G. Danezis and P. Golle, editors, *Privacy Enhancing Technologies*, volume 4258 of *Lecture Notes in Computer Science*, pages 36–58. Springer, 2006.

[2] Android Developer's Guide – Manifest Permissions. `http://developer.android.com/reference/android/Manifest.permission.html`.

[3] Android Market. `https://market.android.com`.

[4] AppBrain. `http://www.appbrain.com`.

[5] D. Barrera, W. Enck, and P. C. van Oorschot. Seeding a Security-Enhancing Infrastructure for Multi-market Application Ecosystems. Technical report, Carleton University, April 2011. TR-11-06.

[6] D. Barrera, P. C. van Oorschot, and A. Somayaji. A Methodology for Empirical Analysis of Permission-Based Security Models and its Application to Android Categories and Subject Descriptors. In *Proc. of the 17th ACM conf. on Computer and Communications Security*, CCS '10, pages 73–84. ACM, 2010.

[7] J. Bonneau, J. Anderson, and L. Church. Privacy suites: shared privacy for social networks. In *Proc. of the 5th Symposium on Usable Privacy and Security*, SOUPS '09. ACM, 2009.

[8] P. H. Chia, A. P. Heiner, and N. Asokan. Use of ratings from personalized communities for trustworthy application installation. In *Proc. of the 15th Nordic conf. in Secure IT Systems*, NordSec '10, 2010.

[9] P. H. Chia and S. J. Knapskog. Re-evaluating the wisdom of crowds in assessing web security. In G. Danezis, editor, *Financial Cryptography and Data Security, FC '11*, volume 7035 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 2012.

[10] F. Cohen. Computational aspects of computer viruses. *Computers & Security*, 8(4):297–298, 1989.

[11] F. J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7:171–176, March 1964.

[12] W. Enck, M. Ongtang, and P. McDaniel. On lightweight mobile phone application certification. In *Proc. of the 16th ACM conf. on Computer and Communications Security*, CCS '09, pages 235–245. ACM, 2009.

[13] Facebook Developers – Permissions. `https://developers.facebook.com/docs/reference/api/permissions/`.

[14] Facebook partners with WOT. Article on ArcticStartup website, May 2011. `http://www.arcticstartup.com/2011/05/12/facebook-partners-with-wot-to-protect-its-700-million-users`.

[15] A. P. Felt. Personal Communication.

[16] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In *Proc. of the 18th ACM conf. on Computer and Communications Security*, CCS '11, pages 627–638. ACM, 2011.

[17] A. P. Felt, K. Greenwood, and D. Wagner. The effectiveness of application permissions. In *Proc. of the 2nd USENIX conf. on Web application development*, WebApps '11. USENIX Association, 2011.

[18] Google Chrome Extensions – Permission Warnings. `http://code.google.com/chrome/extensions/permission_warnings.html`.

[19] Google Chrome Web Store – Extensions. `https://chrome.google.com/webstore?category=ext`.

[20] J. King, A. Lampinen, and A. Smolen. Privacy: Is there an app for that? In *Proc. of the 7th Symposium on Usable Privacy and Security*, SOUPS '11, pages 12:1–12:20. ACM, 2011.

[21] K. Kostiainen, E. Reshetova, J.-E. Ekberg, and N. Asokan. Old, new, borrowed, blue –: a perspective on the evolution of mobile platform security architectures. In *Proc. of the 1st ACM conf. on Data and Application Security and Privacy*, CODASPY '11, pages 13–24. ACM, 2011.

[22] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[23] M. Marsall. How HTML5 will kill the native app. Article on VentureBeat website, April 2011. `http://venturebeat.com/2011/04/07/how-html5-will-kill-the-native-app/`.

[24] T. Moore and B. Edelman. Measuring the perpetrators and funders of typosquatting. In R. Sion, editor, *Financial Cryptography and Data Security, FC '10*, volume 6052 of *Lecture Notes in Computer Science*, pages 175–191. Springer, 2010.

[25] Our project site. `http://aurora.q2s.ntnu.no/app`.

[26] Socialbakers – Applications on Facebook. `http://www.socialbakers.com/facebook-applications`.

[27] J. Tam, R. W. Reeder, and S. Schechter. I'm Allowing What? Disclosing the authority applications demand of users as a condition of installation. Technical report, Microsoft Research, 2010. MSR-TR-2010-54.

[28] Watir – Web Application Testing in Ruby. `http://watir.com`.

[29] Web of Trust (WOT). `http://www.mywot.com`.

[30] WhatApp? A Stanford Center for Internet and Society website. `https://whatapp.org/`.

[31] D. M. Wilkinson. Strong regularities in online peer production. In *Proc. of the 9th ACM conf. on Electronic commerce*, EC '08, pages 302–309. ACM, 2008.