

Geolocation Privacy and Application Platforms

Position Paper

Nick Doty
School of Information
UC Berkeley
npdoty@ischool.berkeley.edu

Erik Wilde
School of Information
UC Berkeley
dret@berkeley.edu

ABSTRACT

Security and privacy issues for *Location-Based Services (LBS)* and geolocation-capable applications often revolve around designing a *User Interface (UI)* such that users are informed about what an application is doing and have the ability to accept or decline. However, in a world where applications increasingly draw on a wide variety of LBS providers on the back-end, and where more and more applications are using small-screen or even screenless devices, UI-centered views of designing security and privacy are no longer sufficient. In this position paper, we describe the increasingly varied landscape of platforms that users face today and the privacy issues of each. We argue for a service-oriented approach, so that security and privacy issues are described and negotiated in a machine-readable way, and can thus be adapted to new platforms and UIs more easily. While *User Experience (UX)* is important, we believe it should be derived from a service-oriented view, instead of being designed for each platform individually.

Categories and Subject Descriptors

K.4.1 [Computers and Society]: Public Policy Issues—*Privacy*

General Terms

Design, Human Factors, Standardization

Keywords

Location-Based Services, Privacy Policies, Geolocation

1. INTRODUCTION

The proliferation of devices that geolocate themselves has raised dramatic concerns about user privacy along with the potential advantages of location-based services. One important dimension to evaluating the privacy of location-based services is the variation across different platforms: users

encounter geolocation on dedicated navigation devices, in native software installed on laptops, desktops and mobile phones, in GPS-enabled cameras, in networked smart objects, and generally on the Web. The growing popularity of “app stores” raises important questions about the different levels or expectations of trust that users have with software and services from different platforms and provenances.

Furthermore, privacy does not have a clear definition the way we might provide one for security. Instead, we find in privacy issues of appropriateness, control, transparency, notice, and consent, and the importance of limitations on distribution, retention, secondary use, and aggregation [3]. As a result, the situation of privacy on different platforms depends not only on the technical implementation details of those platforms, but also on the platform designers’ interpretation of what privacy means and what protections should be supported and communicated by the platform itself. We see, therefore, a wide range in how each of these ten factors is supported by different platforms, a brief survey of which we provide below.

The recent surge in popularity of platform-specific applications (initiated by the popular iPhone and its associated App Store) highlights that users increasingly are faced not only with more services that use location as an essential part of the service experience, but also with services that can be consumed by users via different platforms. From the user’s point of view, it should be possible to understand and meaningfully control important privacy issues regardless of the concrete platform. In this position paper, we argue that geolocation privacy (like many other policy issues) should thus be approached not only on a user interface level but also on a service design level, so that the privacy control can be designed and implemented consistently across various platforms.

While our discussion in this paper focuses on smartphones as a popular and rapidly growing landscape of application platforms which highlights the problems of a fragmented platform landscape, our goal is to make the point that one should look *past* specific platforms and UIs to understand user privacy. While smartphones still retain certain aspects of desktop-oriented applications (they have screens and provide ways for user interaction through keyboards or touchscreens), the next wave of applications promises to be more pervasive and integrated into the physical world [6], where it becomes obvious that privacy controls may even have to be negotiated in UI-less scenarios, where user agents transparently exchange machine-readable privacy policies and then decide what information to share.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SPRINGL 2010, November 2, 2010, San Jose, California, USA.
Copyright 2010 ACM 978-1-4503-0435-1/10/11 ...\$10.00.

2. APPLICATION PLATFORMS

For the purposes of this paper, we mostly look at platforms for mobile (i.e., smartphone) applications. While this looks only at a part of the spectrum of possible user-facing platforms, it is diverse enough to highlight the different capabilities and affordances of specific application platforms. Each of these platforms has different capabilities for exposing and controlling access to location information on the device, and most platforms also have specific UI design patterns and best practices which make it hard to define privacy controls on the UI level which would meaningfully work across different platforms.

The following list is not complete, but represents a majority of platforms which are currently deployed for mobile applications. While these platforms are experienced as runtime environments on specific devices, it is important to distinguish between the platform specification and the platform implementation (Section 2.1).

- *iOS*: Previously known as *iPhone OS*, this platform started the recent popularity of native mobile application platforms. With version 4 (released mid-2010), the platform has moved geolocation controls into the UI chrome, and other features of the platform (such as the ability of applications to run in the background based on location information) have been location-based as well. Currently, iOS is the most popular platform for native smartphone applications.
- *Android*: Google's platform has seen a recent surge in popularity and has a more open model of platform availability: the platform's code is open source, and has been adopted by a variety of mobile phone vendors. In contrast to iOS, many manufacturers have chosen to augment the base platform with their own applications and UI. Android also has fewer design guidelines for UI design and development, which results in a less unified user experience.
- *BlackBerry*: While BlackBerrys are widely deployed, the implementation platform so far has not been overly popular; most BlackBerry users rely on built-in applications. BlackBerry will start supporting modern HTML5 technologies late in 2010; current versions of the included Web browser are very rudimentary in their support for modern Web standards.
- *Windows Phone 7*: Microsoft's platform is not yet released, but will likely have some success in the mobile device market. Once the platform and first devices are available in late 2010, it will be easier to predict the possible success of the platform, and thus its possible influence on the platform landscape.
- *MeeGo*: This Linux-based platform is a merger of Nokia's *Maemo* and Intel's *Moblin* and is expected to gradually replace Nokia's *Symbian* platform. The announcement and release of MeeGo happened early in 2010, and so far it is unclear whether this platform will gain significant market share.
- *WebOS*: Palm's platform is interesting in that it uses *JavaScript* as the native platform language. However, its set of APIs and features as well as the UI are proprietary, so while in principle this platform could adopt

Web-level APIs more easily than others (because of the underlying JavaScript language), in practice it will likely remain a proprietary platform. Because of the limited success of WebOS devices thus far, it is not currently clear whether this platform will gain significant market share.

- *Symbian*: This older platform is still the main platform for Nokia products and is the most deployed mobile platform worldwide. However, few Symbian users use it as a platform for installing applications, it is mostly used as a fixed platform with a limited set of built-in applications.
- *Java 2 Micro Edition (J2ME)*: J2ME was the first mobile runtime platform that was available across a substantial variety of devices. It was plagued with implementation differences and difficult deployment procedures for applications, but still is available on the less-capable phones below the smartphone level, often referred to as *feature phones*. Recently, *JavaFX* has been proposed as a possible successor of J2ME, but so far there seems to be little uptake of this platform on mobile devices.
- *HTML*: Web technologies have had great success as a platform for desktop computers, but their uptake in mobile settings has been overshadowed by the recent success of native applications. However, the loose set of technologies collectively referred to as *HTML5* have a lot of potential to repeat the success of Web technologies in the mobile sector: the browser-based Geolocation API [7], for example, has been the first Web-based location-oriented technology that is close to becoming a stable standard and mobile platforms already show good support for this API in their built-in browsers.

This list of platforms summarizes the most popular platforms available today, but it is incomplete. For the purpose of this paper, our goal is not to present a complete list of platforms, but to highlight the fact that the market for mobile application platforms is highly fragmented and will likely remain so for the foreseeable future. Apart from this variety of available platforms, it is informative to look at the difference between platform specifications and platform implementations (Section 2.1), as well as the different models for how applications on these platforms are deployed and installed (Section 2.2).

2.1 Platform Specifications and Versions

In principle, the above list does not consist of actual platforms (i.e., products implementing the actual platform), but specifications of runtime environments. In some cases, there is only one implementation of such a platform (e.g., there is only one iOS and this is controlled and developed by Apple, and competing implementations will not exist), whereas in other cases there may be multiple products implementing the same platform specification (e.g., HTML is implemented by a number of browsers) and ideally these products are all capable of running applications built for that platform.

For open platforms (i.e., platform specifications which allow multiple implementations), new platform implementations may add new functionalities, APIs, or UIs, and thus assumptions about a certain set of APIs or UIs may not be

consistent across different implementations of a single platform specification. As one example, Android has seen some uptake for resource-constrained devices, because it is available open source and can conveniently serve as a platform for deployment on those devices. However, those devices may have different constraints on device capabilities and UI framework, in which case a purely UI-oriented strategy for providing access to privacy information and other important policies might not be appropriate. The *Nook* e-book reader, for example, runs the Android operating system, but because of its dedicated uses and interface technologies has different affordances from smartphones, Android’s original target devices.

Also, as platforms develop, details about APIs or UIs change between versions. For example, while the iOS platform implemented a set of user controls around geolocation access for applications in prior versions, this changed with the recently released version 4 which added a more visible sign (in the form of a small compass arrow in the status bar of the screen) of an application’s access to geolocation information. While platform versioning has most often been discussed in the context of HTML, it will likely become an issue for other mobile platforms as well, as devices with older versions of the platform remain deployed and are not consistently updated to newer versions throughout the lifetime of the device. Version updates may not even be a viable choice for older devices, which may not have the resources to run a newer and often more heavyweight version of a platform.

2.2 Application Deployment and Installation

One of the major differences between platforms is how applications are deployed and installed. While the Web’s deployment model centers around runtime installation (the code of a Web application is downloaded when it is used), all other platforms have more static models, where applications have to be installed ahead of time and then are permanently available on a device.

While the Web’s current runtime-only model is unique among the platforms we’ve considered and entails different ways in which code installation and execution is handled, a variety of new functionalities often loosely referred to as “HTML5” now make it possible to implement and deploy offline Web applications [10], such that the Web is itself an offline-capable platform.¹ Once HTML5 becomes a stable platform, the Web’s unique ability to combine online and offline applications will require new ways of handling a wide variety of user control issues, including controlling access to device capabilities (popular examples are GPS and camera) and exposing policies around the data online services receive.

One argument in favor of the offline deployment model is that users can have a certain level of trust in installed applications, either because they are delivered through a centralized marketplace (Section 5), or because the installation package is bundled and cryptographically signed and thus is more trustworthy than the more fluid model of resource access on the Web. While the Web does have a security model on the level of Web resources (through secure connections, server-side certificates, and access policies enforced by browsers), it is more difficult to enforce on the application level where a single application may include an intricate

¹Another approach of using the Web as an offline-capable application platform is *ChromeOS*, a Linux-based mobile OS that only supports browser-based applications.

set of interconnected resources. However, even though the deployment and installation process of non-Web platforms is typically static, the ability of applications to connect to the Web and online services may create additional security and privacy exposures which are outside the scope of the application installation process.²

3. GEOLOCATION PRIVACY ON THE WEB

The W3C Geolocation API is an increasingly popular standards-based alternative to multiple fledgling proprietary alternatives for JavaScript access to device geolocation information. Currently a “last call working draft”, the API has already been implemented in release versions of Mozilla Firefox, Google Chrome and Apple Safari and Mobile Safari, and in experimental versions of Opera and Microsoft Internet Explorer. Although the standard has not been finalized, more than a third of Web users are using supporting browsers³ and our ongoing research finds that at least hundreds of thousands of Web pages use the API. Recent versions of iOS and Android browsers also support the API, suggesting that the W3C Geolocation API is also widely used on the mobile Web.

The W3C Geolocation Working Group chose not to include privacy fields or hooks in the API itself after considerable (and ongoing) debate. Instead, the specification includes normative language describing how sites should provide clear and conspicuous disclosures to their users and requires browsers to get explicit yes-or-no permission from users before sharing their latitude and longitude coordinates. Research on the privacy practices of Web sites that use the draft API show significant variation in what users encounter but find that virtually none are informed of all the relevant information before being prompted for their location [3], raising concerns about use, distribution and retention and the absence of truly informed consent. From a services-focused approach, privacy is not just an issue of trust between a user and a single Web site, but a question of understanding of how personal information will be used, re-used, shared or published by the Web site, its partners or other services “down the line”.

Research on the privacy practices of browsers that implement the draft W3C Geolocation API show the variation users encounter in how Web browsers prompt for user consent, remember permissions or allow revocation, raising concerns about user control [1]. Furthermore, the small number of underlying location providers (provided by Google, Skyhook Wireless and Apple) that browsers use when the device doesn’t have access to GPS location raises the risk of aggregation. Giving your location to several different sites over the course of a day also gives your location every time to the same location provider: a distinction not readily apparent to the uneducated user.

The arguments around the W3C Geolocation API have often focused on the question of trust or the lack thereof on the Web. Can we allow site-specified promises about

²As one example, Apple tries to limit this risk by disallowing iOS applications from dynamically loading and executing code, but this does not prevent applications from connecting to Web services and creating security or privacy loopholes this way.

³Aggregate of Firefox 3.6, 3.5, Chrome 5 and Safari 5 market share percentage is 38.1% as of June 2010 according to StatCounter.com.

privacy practices to be put forth by the Web browser? Can we trust Web sites to make their policies clear to users before beginning to track their locations?

Debate continues within the W3C community over how best to handle privacy for Geolocation and other “advanced Web APIs”, including an effort from the *Device APIs and Policy (DAP) Working Group*⁴ to unify and consolidate different experiences across different APIs. There are various proposals to better support geolocation privacy either by enabling users to express their own privacy preferences in a machine-readable way [2] and embed privacy controls in resources as they’re shared between services [4] or by allowing web sites to unambiguously represent their privacy policies in simultaneously machine-readable and human-recognizable formats [8].

4. GEOLOCATION PRIVACY ON THE DESKTOP

Recent work from both Microsoft and Apple has led to standardized OS-level frameworks for a large install base⁵ of Windows and Mac OS users with an abstracted geolocation API accessible to installed native software. As laptops and netbooks become increasingly common and increasingly mobile, traditional “desktop” operating systems can be an important platform for location-based services.

Windows 7 and Mac OS X Snow Leopard take substantially different approaches to privacy protections around geolocation. The Windows 7 Location Platform is off by default, but once turned on is available to all applications running on the machine as a particular user (out of concern that “data exposed to one process may not be secure from access by another process”⁶). When an application does access the current location, the user is notified in the task bar and records of which application accessed the user’s location when are kept in the system event log for later inspection.

In contrast, applications that request the device location in Mac OS X 10.6 prompt the user for per-application yes-or-no permissions (similar to the current situation on the Web). However, the operating system provides no notification when an application is accessing the computer’s location and no logs are kept for later inspection.

Installed applications on the desktop certainly have a different level of trust than a completely uncontrolled Web site and may be more likely to have an existing trust relationship with the user (the OS, for example, may warn users when they run a new application from an untrusted source). Nevertheless, it remains an open question whether users understand how these desktop applications use, store or re-distribute their location information. Since for many use cases the advantage of location access for a “desktop” application relies on accessing online services, supporting geolocation privacy on the desktop depends on the privacy practices of those external services.

5. APPLICATION STORES

Of growing popularity are application stores, or “app stores”, which provide centralized locations for discovering, purchas-

⁴<http://www.w3.org/2009/dap/>

⁵Over 15% of desktop operating system market share as of July 2010 according to NetMarketShare.com.

⁶[http://msdn.microsoft.com/en-us/library/ff545686\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ff545686(VS.85).aspx)

ing, installing and reviewing software for various platforms. Popularized by Apple’s iTunes App Store for finding applications for iOS devices, app stores have sprung up for Palm, Android, Nokia, Windows Mobile and other devices.⁷

Application stores, because of their centralized nature, provide two interesting side effects for expressions of privacy on a particular platform. First, stores can define standards for metadata (potentially including privacy policies), require that such metadata be present and provide a consistent way of presenting such metadata to users.

Second, unlike the decentralized Web or the traditional model of installing software from any location, application stores provide a framework within which certain rules can be defined and enforced. Store owners can evaluate and approve software, making a promise to users that the software meets certain standards; as a result, users may be more comfortable trusting any particular application available through the store.

Below, we look at two examples of mobile device app stores and also a potential future example combining app stores with the Web.

5.1 The iTunes App Store

What level of trust users have in software from Apple’s very popular *App Store* remains an open question. Apple is famous (infamous, in some circles) for their vetting and inspection of “apps” before release to the public. Still, the platform does not assume complete trust: when an iOS application requests the device’s location, a modal yes-or-no dialog is first presented to the user.

Although Apple’s included Camera app on the iPhone includes usage text as part of its location prompt (“Photos will be tagged with the location where they were taken.”), this functionality is neither required nor even available to external iOS developers. Still, Apple puts normative legal requirements on how developers can use location with the platform. For example, using location only for advertising purposes is explicitly prohibited,⁸ though the iAd platform allows for location-based ads.⁹ Research has suggested that even with Apple’s vetting process, malicious applications distributed through the App Store could gain access to user’s personal information, including current and past locations [9].

5.2 The Android Market

The Android platform allows multiple app stores, but the *Google Android Market* is the biggest one available. Installing an Android application requires first accepting a list of device permissions for that particular application. It remains an open question whether users understand all the implications of the variety of permissions. Though users are provided up front with a list of capabilities each application will have, application developers are not required to describe how personal information (like the user’s location) will be used, whether it will be stored and to which services or third parties it will be re-distributed.

Underlining the concern, recent news of a popular Android

⁷For a long but likely incomplete list, see http://en.wikipedia.org/wiki/List_of_digital_distribution_platforms_for_mobile_devices

⁸<http://developer.apple.com/iphone/news/archives/2010/february/#corelocation>

⁹<http://advertising.apple.com/>

application that “phones home” with personal details from the device has raised concerns about privacy on Android.¹⁰ Ongoing research suggests the possibility of determining automatically at runtime when applications are accessing and retransmitting personal information.¹¹

Google is expected to take a lighter touch with vetting and removing applications from the store than Apple’s controversial review process. How the differences between the application stores will translate into differences in user trust or comfort around privacy issues remains to be seen.

5.3 A Web App Store?

Of particular recent interest is movement towards a *Chrome Web Store*,¹² which would seek to improve discoverability and monetization of otherwise very distributed Web applications. Announced at Google’s May I/O Conference¹³ and proposed as an answer to the problem of permissions “bundling” [5], the Web Store would allow users to install Web applications to their browser home page and would provide the same unified discovery and purchase experience seen in mobile device app stores. Google also believes that the Web Store can provide an up-front UI for listing the necessary permissions for a particular application, rather than require separate opt-in requests for access to location, the file system, contacts listings, camera and other device access APIs. There is, however, a fundamental difference between the usual static installation process on native platforms, and the essentially online nature of Web applications; even “installed” Web applications can be updated at any time, and the Web’s fundamental principle of resources and caching demand that such an update request is granted by the browser.

Presence in an application store could provide an added level of enforcement not usually present with Web applications: Google or other store administrators could remove applications that violate certain policy rules. Still, it remains unclear whether the Web Store will allow or require applications to describe their policies at the same time they request permissions.

6. CONCLUSIONS

Users who encounter location-enabled services on these different platforms encounter different user experiences, different technical capabilities, and different levels of trust.

We disagree with the idea that there are “good guys” and “bad guys” on the Web or writing software in general, and though trust and identity may be keys to security, they are insufficient to robustly protect user privacy. Nevertheless, the variety of platforms and their models for privacy does allow for a valuable comparison. We do notice a consistent deficiency across platforms in providing policy information that would allow for truly informed consent around privacy issues of use, distribution or retention. This problem is all the more extreme when seen from a services point of view, where location information is passed between multiple end-points and users may encounter the same service from a variety of applications on different platforms and with different

UI affordances.

We see this survey of geolocation privacy on different platforms as just a starting point for our research. We believe there is work to be done in surveying users’ actual experiences with these different platforms: to what extent do user mental models of device and platform comport with the actual practices of APIs and software? In the long run, we hope privacy practices can be unified through machine-readable descriptions of policy in order to better inform users and provide more consistent and coherent representations of policy across devices and platforms.

7. REFERENCES

- [1] MARCOS CACERES. Privacy of Geolocation Implementations. In W3C Workshop on Privacy for Advanced Web APIs [11].
- [2] ALISSA COOPER, JOHN MORRIS, and ERICA NEWLAND. Privacy Rulesets: A User-Empowering Approach to Privacy on the Web. In W3C Workshop on Privacy for Advanced Web APIs [11].
- [3] NICK DOTY, DEIRDRE K. MULLIGAN, and ERIK WILDE. Privacy Issues of the W3C Geolocation API. Technical Report 2010-038, School of Information, UC Berkeley, Berkeley, California, February 2010.
- [4] THOMAS DÜBENDORFER, CHRISTOPH RENNER, TYRONE GRANDISON, MICHAEL MAXIMILIEN, and MARK WEITZEL. Making Privacy a Fundamental Component of Web Resources. In W3C Workshop on Privacy for Advanced Web APIs [11].
- [5] IAN FETTE and JOCHEN EISINGER. Practical Privacy Concerns in a Real World Browser. In W3C Workshop on Privacy for Advanced Web APIs [11].
- [6] DOMINIQUE GUINARD, ERIK WILDE, and VLAD TRIFA, editors. *First International Workshop on the Web of Things (WoT 2010)*, Mannheim, Germany, March 2010.
- [7] ANDREI POPESCU. Geolocation API Specification. World Wide Web Consortium, Candidate Recommendation CR-geolocation-API-20100907, September 2010.
- [8] AZA RASKIN and ARUN RANGANATHAN. Privacy: A Pictographic Approach. In W3C Workshop on Privacy for Advanced Web APIs [11].
- [9] NICOLAS SERIOT. iPhone Privacy. In *Black Hat DC 2010*, Arlington, Virginia, USA, 2010.
- [10] ANNE VAN KESTEREN and IAN HICKSON. Offline Web Applications. World Wide Web Consortium, Note NOTE-offline-webapps-20080530, May 2008.
- [11] *W3C Workshop on Privacy for Advanced Web APIs*, London, UK, July 2010.

¹⁰<http://blog.mylookout.com/2010/07/mobile-application-analysis-blackhat>

¹¹<http://www2.seattle.intel-research.net/~jjung/>

¹²<https://chrome.google.com/webstore>

¹³<http://code.google.com/events/io/2010/sessions/web-apps-chrome-web-store.html>