Poster: Flex-P: Flexible Android Permissions

Kurt Mueller *(student)* and Kevin Butler *(faculty)*
Department of Computer and Information Science
University of Oregon
Eugene, Oregon 97403-1202
Email: {kurtm,butler}@cs.uoregon.edu

***Abstract*—The Android mobile operating system and its application store, the Android Market, are growing rapidly. Because Google does not curate the Market, the presence and impact of malware in the Market are also growing, and management of device security is left to individual users who are faced with an all-or-nothing decision when installing an app from the Market: either install the app and grant it all the permissions it requests, forever, or cancel the installation. To provide more control to users over the permissions granted to applications, we present Flex-P, a set of additions and modifications to the Android operating system for managing application permissions at install time and at runtime.**

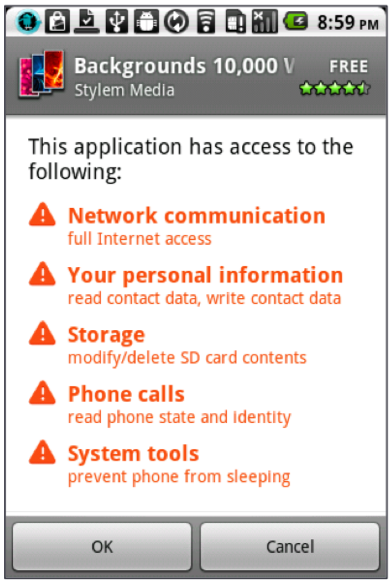
I. Introduction

The Android mobile operating system has gained tremendous market share since its introduction in 2006, and in early 2011 it surpassed Apple's iOS and RIM's BlackBerry platforms to become the most popular smartphone platform in the United States[3]. One component of its growth and success is the Android Market, which makes 150,000 applications available to Android users[2] directly from their mobile devices, and through the Android Market website.

While the Market has been a boon to both Android users and legitimate application developers, it has also provided a new avenue for malware authors to distribute their goods. In contrast to Apple's iTunes App Store, in which submitted applications are individually screened by Apple staff before publication to ensure compliance with App Store rules and to prevent malicious apps from being distributed, there is no pre-publication curation of apps in the Android Market. Programs have been removed from the Market by Google after reports of bad behavior from users[4], though at that point damage may have already occurred.

The permissions system built into the Android OS partially addresses this issue by informing users at the time of application installation of the permissions that an application may use during the course of its normal operation. In order to complete the installation, the user must explicitly grant to the application the permissions it has requested; if the user does not agree to grant such permissions, canceling the application installation is the only option.

Figure 1 shows the permissions request screen displayed during installation of a popular wallpaper/background app from the Android Market. Exactly why the application needs "full Internet access" and the ability to "read contact data, write contact data" is not apparent, and a potential user would be justified in hesitating to install the app, or declining to do so entirely. If the user chooses to proceed with the installation of such an app, they are irrevocably granting the app the rights it has requested.


Backgrounds 10,000 V FREE
Stylem Media

This application has access to the following:

Network communication
full Internet access

Your personal information
read contact data, write contact data

Storage
modify/delete SD card contents

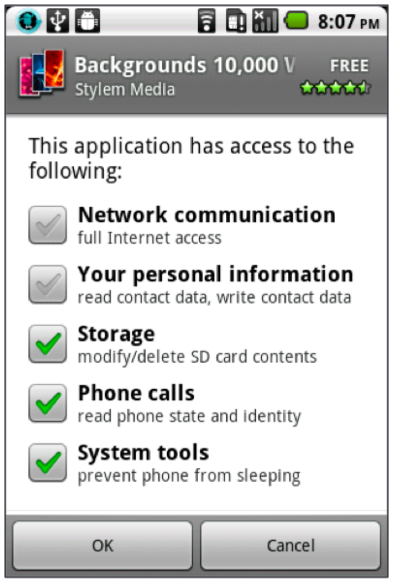
Phone calls
read phone state and identity

System tools
prevent phone from sleeping

Fig. 1. Stock Android Permissions Request

In this poster we present Flex-P, which stands for Flexible Permissions. Flex-P is a set of modifications and additions to the Android 2.2 operating system that gives users more fine-grained control of granted permissions at app installation time, as well as the ability to modify granted permissions at runtime or at any time in the future through Android's Settings feature. Flex-P augments Android's native authorization system to enable checking of Flex-P data structures, and provides a framework for future enhancements to the installation and app management processes.

Previous efforts to improve the security of the Android platform include Kirin[1] and Apex[5]. Kirin implements install-time policy checking to prevent installation of applications that have potentially dangerous combinations of granted permissions, but it does not allow users to change permissions. Apex is similar to Flex-P in that it provides the ability for users to grant or deny application permissions; however, it requires modification of individual permissions (rather than permission groups, as in Flex-P), and adds new restriction types to granted permissions, such as limiting a permission to a certain number



8:07 PM

Backgrounds 10,000 V
Stylem Media
FREE

This application has access to the following:

- ☐ **Network communication** full Internet access
- ☐ **Your personal information** read contact data, write contact data
- ✓ **Storage** modify/delete SD card contents
- ✓ **Phone calls** read phone state and identity
- ✓ **System tools** prevent phone from sleeping

Fig. 2. Flex-P Permissions Request

of uses per day. While this approach is powerful, it may be overly burdensome to end users. We have attempted to strike a manageable balance between control and usability with Flex-P.

II. Flex-P approach

A. Displaying permissions during installation

In Flex-P, we have modified the Android `PackageInstallerActivity` and related classes to display an application's requested permissions (as described in `AndroidManifest.xml`) with checkboxes next to each permission group, allowing the user to grant or deny individual groups of permissions. The modified installer screen is shown in Figure 2, with some permission groups granted and others denied. Flex-P stores user choices in its own file format, overcoming limitations imposed by the static `AndroidManifest.xml` file.

B. Modifying permissions after installation

It is possible with Flex-P to change the permissions granted to an application after installation, at any arbitrary time, through the built-in Android Settings / Applications / Manage Applications tool. Flex-P includes a modified `InstalledAppDetails` class and supporting classes to display the permissions for an application, with checkboxes and a `Save` button, to enable editing and saving of permissions settings.

C. Checking permissions at run-time

The stock Android authorization system simply checks an application's `AndroidManifest.xml` file at run-time when the application tries to perform an activity that is restricted. Since Flex-P stores enhanced information about which permissions have been granted, it is necessary to also check the application's `.flexp` file during the authorization process. The name of the permission being checked and the `UID` of the requesting application are passed to Flex-P, and if Flex-P has a record of the application in its permission store then it checks the application's `.flexp` record and returns the `granted` boolean value contained therein.

III. Evaluation

Performance of a stock Android installation was compared with a modified, Flex-P-enabled installation. For the regular Android installation, without Flex-P, the average startup time over five trials was 106.8 seconds. With Flex-P installed, the average startup time over five trials was 114.6 seconds. The added overhead of initializing the Flex-P system and calling the Flex-P static authorization method added 7.8 seconds, or approximately 7.3%, to startup time. Given that this increase includes the one-time Flex-P initialization sequence, it is likely that ongoing Flex-P authorization checks after initialization will have a smaller impact on performance. Subjectively, there seems to be no difference in responsiveness between a Flex-P enabled installation and one with stock Android authorization.

IV. Conclusion

In the absence of strict Android Market curation, like in Apple's App Store, it is up to users to manage the security of their devices within the constraints established by the mechanisms built in to the Android operating system. Though Android provides an authorization system to keep applications from overstepping their bounds, it is inflexible both at app installation time and afterward. Flex-P provides an initial framework for giving more control over application permissions to users when they install applications, while respecting the established goal of keeping the installation interface simple and not confusing or overwhelming users with choices. It also gives users the ability to change granted permissions at any time after app installation. Future work is focused on maintaining application robustness in the face of revoked permissions, and on providing the ability for application developers to specify some permissions as required and others as optional while justifying their use of potentially dangerous permissions.

References

[1] William Enck, Machigar Ongtang, and Patrick McDaniel. On lightweight mobile phone application certification. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 235–245, New York, NY, USA, 2009. ACM.

[2] http://androinica.com/2011/02/android-has-150k-apps-350k-daily-activations-and-more-notes-from-eric-schmidts-mwc keynote/. Android has 150k apps, 350k daily activations, and more notes from eric schmidt's mwc keynote.

[3] http://blog.nielsen.com/nielsenwire/ online_mobile/who-is-winning-the-u-s-smartphone battle/. Who is winning the u.s. smartphone battle?

[4] http://googlemobile.blogspot.com/2011/03/update-on-android-market security.html. An update on android market security - official google mobile blog.

[5] Mohammad Nauman, Sohail Khan, and Xinwen Zhang. Apex: extending android permission model and enforcement with user-defined runtime constraints. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 328–332, New York, NY, USA, 2010. ACM.