AdDroid: Privilege Separation for Applications and Advertisers in Android

Paul Pearce, Adrienne Porter Felt
Computer Science Division
University of California Berkeley
{pearce, apf}@cs.berkeley.edu

Gabriel Nunez
Sandia National Laboratory*
gnunez@sandia.gov

David Wagner
Computer Science Division
University of California Berkeley
daw@cs.berkeley.edu

ABSTRACT

Advertising is a critical part of the Android ecosystem—many applications use one or more advertising services as a source of revenue. To use these services, developers must bundle third-party, binary-only libraries into their applications. In this model, applications and their advertising libraries share permissions. Advertising-supported applications must request multiple privacy-sensitive permissions on behalf of their advertising libraries, and advertising libraries receive access to all of their host applications' other permissions. We conducted a study of the Android Market and found that 49% of Android applications contain at least one advertising library, and these libraries overprivilege 46% of advertising-supported applications. Further, we find that 56% of the applications with advertisements that request location (34% of all applications) do so only because of advertisements. Such pervasive overprivileging is a threat to user privacy. We introduce *AdDroid*, a privilege separated advertising framework for the Android platform. AdDroid introduces a new advertising API and corresponding advertising permissions for the Android platform. This enables AdDroid to separate privileged advertising functionality from host applications, allowing applications to show advertisements without requesting privacy-sensitive permissions.

1. INTRODUCTION

The Android Market [2] has given Android developers an avenue to rapidly develop and distribute applications directly to consumers. Mobile advertising services such as AdMob [1] and Millennial Media [21] play a key role in this ecosystem by allowing developers to generate revenue from advertisements. These advertisers distribute libraries that make it easy for applications to request advertisements and display them in their user interfaces.

*Work done while at the Computer Science Division, University of California Berkeley.

ASIACCS '12, May 2–4, 2012, Seoul, Korea.

Before developers can distribute an Android application, they must identify and declare what permissions their applications need. The installation process shows these permissions to users if they choose to install the application. However, the Android security model does not provide any special support for advertisements. This means that there is no way for the user or the operating system to differentiate between the permissions necessary for the host application and those needed by advertising libraries. A simple advertisement-supported application (e.g., a game or custom ringtone application) must request multiple privacy-sensitive permissions unrelated to its functionality just so it can display advertisements. Advertising libraries often require permissions for network connectivity, location information, and phone state.[1] Since developers include the advertising library as part of the application, applications can use the permissions needed for advertising libraries, and vice versa. This is an instance of *overprivileging*: the advertising libraries require applications to take on additional privileges that they otherwise might not need. Moreover, these additional permissions have serious implications for user privacy.

The current advertising model results in the following classes of threats:

- Advertisers may track users without their knowledge.
- Untrusted applications may have access to unnecessary private data, such as the user's location.
- If an adversary exploits a buggy application, the benign-but-buggy application may inadvertently leak or expose sensitive information to attack.
- Malicious or opportunistic advertising networks could abuse the permissions of their host applications.
- Applications frequently requesting sensitive permissions, or permissions unrelated to the application's functionality, can train users to ignore permission warnings [12].
- Advertising libraries have been observed downloading code over HTTP and dynamically executing it at runtime [13].

To combat these threats we propose *AdDroid*, an extension of the Android platform that provides special support

[1]Applications need phone state to obtain the phone's International Mobile Equipment Identity (IMEI). Developers can use the IMEI as a unique customer identifier.

for advertisements. In AdDroid, the host application and the core advertising code run in separate protection domains. We accomplish this privilege separation by moving privileged functionality into a new Android system service. Application developers may incorporate advertisements into their application by invoking the AdDroid advertising application programming interface (API). Applications using AdDroid no longer need privacy-sensitive permissions just to display advertisements.

To understand the potential value of AdDroid, we analyzed a set of 964 real-world Android applications collected from the Android Market. We found that 49% of applications contain at least one advertising network library. We also found that application developers overprivileged 46% of advertising-supported applications (23% of all applications) by one or more permissions as a direct result of their use of advertising libraries. Further, we found that 27% of advertising-supported applications receive both Internet connectivity and location information, even though the application needs neither for its functionality. This particular combination of permission overprivilege is potentially sensitive because it allows applications to track a user's location without their knowledge. Lastly, we found that 34% of all applications (56% of advertising-supported applications) that request location do so only because of advertisements. With AdDroid, 27% of advertising-supported applications would no longer need Internet access, 25% would no longer need location information, and 8% would no longer need phone state information. Overall, AdDroid would reduce overprivileging in 46% of advertising-supported applications.

Contributions. The contributions of this work include:

- We identify and detail the problem of advertising-related overprivileging, including a detailed model of the threats faced by users and developers (Section 2).
- We design and implement AdDroid, a solution to the threat of advertising overprivilege that applies privilege separation to advertising libraries (Sections 3 and 4).
- We measure the threat of advertising overprivilege and the impact of AdDroid with an advertising and overprivilege measurement study performed on applications from the Android Market (Section 5).

2. BACKGROUND

To frame the problem, we describe the Android permission system and explain how mobile advertising currently works. We then explain how the current system leads to privacy and security threats from legitimate advertising networks, grayware, benign-but-buggy applications, malicious advertising networks, and vulnerable advertising networks.

2.1 Android Permissions

Android employs an install-time permission system to alert users to the privacy and security ramifications of installing an application. A *permission* gives an application the ability to perform a specific privacy- or security-relevant action. For example, the `INTERNET` permission controls communication with remote servers, the `RECORD_AUDIO` permission provides the ability to record audio with the microphone, and the `ACCESS_FINE_LOCATION` permission permits access to the phone's GPS location. An application cannot perform a given operation unless it has the related permission.

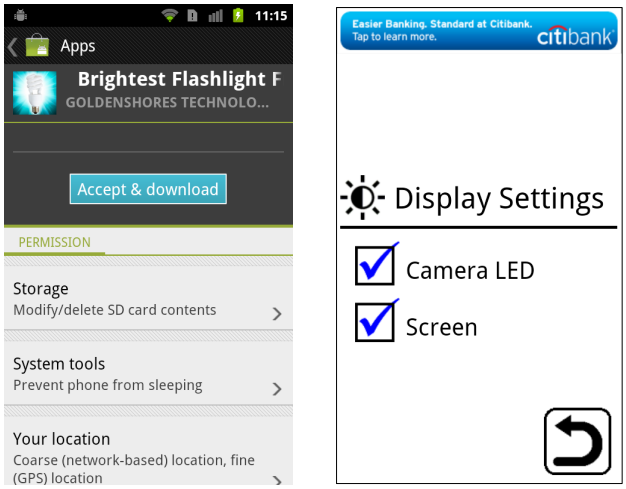

Figure 1: At left, an application's requested permissions are shown as part of the installation process. At right, an example of an advertisement (top of screen) in an application.

In order to gain permissions, applications must first request them. Developers determine (e.g., via testing) the permissions that their applications require, and then they bundle their applications with the appropriate list of permission requests. Different applications request different sets of permissions, based on their functionality.

The Android installer asks users to grant these permission when they install the application. After initiating the installation process, the system shows the user the list of permissions that the application requests. Figure 1 shows a screenshot of the permission authorization process in the Android Market. The user implicitly accepts the permission requests by choosing to complete the installation process. Permission authorization is all-or-nothing; the user must accept all of the application's permission requests or cancel installation. Once installed, these permissions will be available to the application each time it executes.

2.2 Advertising Libraries

Advertising networks serve as middlemen between application developers and advertisers. The advertising networks supply developers with advertising libraries, and developers place the libraries in their applications. The libraries provide APIs for inserting advertisements into the application's user interface and handle the fetching, rendering, and tracking of advertisements. These libraries are responsible for 65%-75% of energy usage in free applications [22].

In Android all parts of an application—including the advertising libraries—have the same permissions. This has two consequences. First, an advertising library can use all of the permissions that its host application requests. Second, an application that includes an advertising library must request all of the permissions that the advertising library requires. Figure 2 shows the relationship between

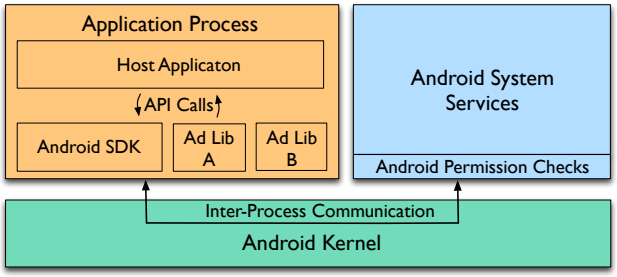

Figure 2: Diagram showing how advertising currently works in Android. The example application uses two advertising networks. Each advertising library access the privileged Android system services via IPC. The Android system does not distinguish between different parts of an application.

advertising libraries, applications, and the Android permission system. Android system services perform privileged operations. Applications access these services via inter-process communication (IPC), and the system services will only perform operations that applications have permissions for. While Android system services are able to perform operations unconstrained by Android application, they do not run as "root."[2]

Certain permissions are commonly associated with advertising libraries. Advertising libraries load advertisements from remote servers, which requires the `INTERNET` permission; they may also use the `ACCESS_NETWORK_STATE` permission to determine whether the phone has access to the Internet. To support location-specific advertisements, applications must request one of the `LOCATION` permissions (either GPS or network location). Some advertising networks use the `READ_PHONE_STATE` permission to obtain the phone's IMEI, which can serve as a unique customer identifier. Advertising networks publish documentation that instructs developers to include these permissions in their lists of required permissions.

2.3 Threat Model

Besides potentially training users to ignore sensitive permission requests [12], five threats arise from the Android advertising ecosystem:

- *Legitimate advertising networks.* Studies suggest that a majority of users have experienced discomfort sharing personal information such as location and browsing history with advertising networks [19, 16, 4]. The purpose of the Android permission system is to inform users about what personal information they will share with a third party, but Android users are currently unable to tell which permissions advertising libraries use. Consequently, users are unaware of what information applications or advertising libraries are sharing with advertising networks and are unable to object to advertising libraries' permission requests. While users can usually identify whether an application contains advertisements, this may only become apparent after they install and use the application. More importantly, users can not tell what information (e.g., location) the application is sending to advertisers.

- *Grayware.* In some situations, users may trust advertising networks more than the application. For example, consumers might be willing to share their location with Google's advertising network but not with unbranded games. Currently, there is no way for a user to identify what data they give to the untrusted application versus what data they give to the advertising network. Consider the case of the unbranded game requesting location permissions. There is no way to determine if the game is taking that location information and using it in some way contrary to the user's privacy, or if a reputable advertising network is using the location to serve location relevant advertisements.

- *Benign-but-buggy applications.* The more permissions a user grants to an application, the more damage it can do if exploited [24]. If an overprivileged application contains bugs, an attacker could make use of these additional privileges to track users, exfiltrate data, and violate user privacy. A buggy application could also unknowingly leak private information it had access too.

- *Malicious advertising networks.* An unscrupulous advertising network could supply application developers with a malicious library that abuses the permissions requested by host applications. Although we are unaware of existing malicious mobile advertising networks, malware has appeared in the Android Market [10], and malicious advertising partners have collaborated with botnets in other settings [20]. A less severe variant of this threat is an *opportunistic* advertising library that uses aggressive, borderline-legal marketing techniques. An example of an opportunistic threat would be a library that collects users' phone contact lists to send spam. Opportunistic advertising libraries that check for and use their host applications' permissions have been seen in the wild [13].

- *Vulnerable advertising networks.* Advertising libraries have been observed downloading code over HTTP and executing it [13]. This behavior could turn a legitimate advertising network into a malicious one if the user is connected to an unsafe wireless network.

In each of these threats, providing applications and advertising libraries with unfettered access to each other's permissions poses significant danger.

3. PROPOSAL

We propose to use *privilege separation* [23] to address the threats outlined in Section 2. In general, privilege separation involves placing the privileged components of the application in a different protection domain from the rest of the application, and assigning a different set of permissions to each domain. In this context, we suggest placing the privileged parts of the advertising library in a separate protection domain, and assigning it a separate set of permissions.

[2]One can think of the Android system services as running with the permissions of a traditional Linux process.

We begin by exploring two traditional solutions for privilege separation: separation within a process and separation between processes. Then, we explain why these solutions are inadequate in the context of Android advertising. We then present an alternative solution, our system *AdDroid*.

3.1 Privilege Separation Within A Process

A natural approach is to introduce a finer-grained permission model, where an application can consist of multiple protection domains, each with its own code and associated set of permissions. It is tempting to assign each Java class to its own protection domain, with all protection domains running in the same virtual machine and process. This would let the advertising library exist in the same process and Java runtime as the rest of the application, but with different permissions. Under this proposal, method calls between the library and application remain unchanged because they are in the same virtual machine, yet the advertising library and application have their own permissions.

Unfortunately, there are two technical problems with this solution. First, some applications include native libraries, which can arbitrarily violate the integrity of the virtual machine running in the same process [9]. Second, the Dalvik virtual machine (which executes Android Java code) does not support isolation between portions of code within the same virtual machine, which would be necessary to prevent one part of an application from interfering with or altering other parts of the application. Java includes a reflection library that explicitly allows other code to violate code encapsulation and integrity running in the same virtual machine. Consequently, we do not think it is feasible to separate the privileges of an application and its advertising library within a Dalvik virtual machine or process.

3.2 Privilege Separation Between Processes

Another approach is for advertising libraries to exist as separate, standalone applications. These advertising applications would run in separate processes from the user application. When a user installs an application with advertisements, the user-selected application would install the appropriate advertising application. User-selected applications would either interact with advertising applications via existing IPC mechanisms, or the Android system would have to be modified to allow the user-selected and advertising applications to share the screen [6]. Advertising applications would consist of the current advertising library and shim code to proxy information to and from the target application. The user-selected and advertising applications would each run with their own sets of permissions. As we envision it, user-selected applications would request a new `ADVERTISING` permission in order to display advertisements, and the system would allow advertising applications to use permissions associated with advertising such as `INTERNET` or `LOCATION`.

Unfortunately, there are numerous hurdles to implementing this approach on the Android platform, including:

- Users who do not want to look at advertisements might uninstall the advertising applications. This would hurt developers and companies that rely on advertising revenue. Application developers expect users to "pay" for free (i.e., subsidized) applications by viewing advertisements.

- The `ADVERTISING` permission is necessary to warn users that the selected application transmits data to an advertising network through an advertising application. However, it would be difficult to enforce this permission. Once a user or host application has installed an advertising application on a phone, other user-selected applications could communicate with it to load advertisements. Advertising applications could refuse to service user-selected applications that lack the `ADVERTISING` permission, but this relies on the advertising applications to enforce the permission requirement. Since advertisers have an incentive to show advertisements to as many applications as possible, they are incentivized to not enforce these permissions. The Android Market could refuse to list advertising applications unless they abide by this constraint, but this would require periodic audits of advertising applications.

- The Android platform would need to be extended to support install-time dependencies between applications so that advertisement-supported applications could list an appropriate advertising application from their preferred advertising network as a prerequisite. The platform would also need to support runtime application dependencies. This would require changes to the Android platform, the Market, and the user interface for application installation. Such changes could present unforeseen abuse opportunities. There is also a potential for reliability and security concerns if the market must manage separate advertising applications for all possible advertising networks. Each phone would need to maintain copies of the versions needed by all of the installed applications. This presents significant logistical obstacles for deployment.

- The Android Market would need to prevent the distribution of impostor advertising networks. An impostor advertising network would collect all of the information sent to it without paying the developer and without the privacy policy of a reputable advertising network.

Consequently, we believe the solution of standalone advertising applications will not suffice. Therefore we present a modified and extended IPC approach which leverages existing infrastructure in the Android platform.

3.3 Our Proposal: AdDroid

We propose AdDroid, a solution which integrates advertising into the Android platform. In this solution, advertising networks do not provide advertising libraries at all. Instead, the Android API is extended to support advertising. The extended Android API relays advertising metadata from applications to advertising networks, fetches advertisements from advertising networks' servers, and handles user interface events. Applications provide the API with configuration data to identify which advertising network(s) to fetch advertisements from, as well as contextual information about the advertisement. This modification to the Android platform introduces a model similar existing system services, and relies on mechanisms already in the platform.

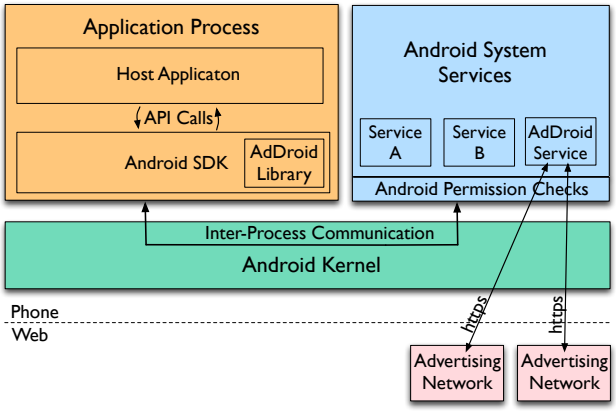

Figure 3: The AdDroid design. Key components include a userspace AdDroid library that adds advertising support to the public API, and a supporting AdDroid system service that communicates with the advertising network's Internet servers.

We also propose the addition of two new permissions: `ADVERTISING` and `LOCATION_ADVERTISING`. Applications wishing to take advantage of AdDroid would request one of these permissions, which would give them access to the new advertising API calls. Advertising-supported applications would not need to request additional permissions for advertising (e.g., `LOCATION`) because all of the privileged advertising-related operations would be performed by AdDroid on behalf of the application. Applications requesting these permissions will not have unfettered access to Internet, location, or phone identifying information. These permissions will *only* grant them access to parts of the advertising API.

Figure 3 shows a system overview of AdDroid. The Android SDK now includes additional advertising-related API calls which are supported by a new system service. All of the advertising-related code that does not require special permissions resides in the Android SDK and does not need to be trusted by the privileged Android components. Advertising functionality that requires permissions (e.g., connecting to the Internet) is implemented in the new service, which can perform privileged operations. The advertising system service is only accessible to applications with the appropriate `ADVERTISING` or `LOCATION_ADVERTISING` permissions. In our proposal, all of the advertising-related code that requires privilege is trusted enough to be part of a system service because it has been written or reviewed by the Android development team. Consequently, it is safe for privileged advertising code to execute as a system service.

This solution mitigates our five threats as follows:

- *Legitimate advertising networks.* The advertising permissions inform users when applications make use of advertising. In particular, the `LOCATION_ADVERTISING` permission informs users when applications include location-aware advertisements. Armed with this information, users can make informed decisions about which applications to install, given their personal privacy preferences.

- *Grayware.* In AdDroid, applications can include advertisements without needing the `LOCATION` or `READ_PHONE_STATE` permissions. This enables users to distinguish between applications that use permissions for their own purposes and applications that use permissions for advertising. For example, a person considering an unbranded game would know that the game developer is collecting his or her information for a purpose other than displaying location-aware advertisements. We expect that our proposal would significantly reduce the number of applications that ask for Internet, location, and identity permissions, which would make the permissions more notable and therefore more effective. If these applications only requested the advertising related permissions, they would have no access to Internet, location, or phone identification information.

- *Benign-but-buggy applications.* Since applications no longer need to request permissions for their advertising libraries, many will have fewer permissions. This means that a buggy application will be less likely to have Internet, location, or phone state permissions, thus reducing the impact of any vulnerability or bug in the application.

- *Malicious advertising networks.* In our model, advertising networks do not provide any code directly to developers choosing to use AdDroid. As such, malicious advertising networks would not be able to leverage application permissions to collect excessive amounts of user data or perform undesirable operations.

- *Vulnerable advertising networks.* Since applications would no longer embed advertising libraries in their applications, advertising libraries cannot introduce vulnerabilities into applications.

As such, we believe AdDroid addresses the privacy and security concerns about advertising identified in Section 2.3.

In addition to improving privacy, AdDroid provides other benefits. Uniting advertising networks under a common API would decrease developer overhead, particularly for applications that use multiple advertising networks. In turn, this gives advertisers more potential customers, as every developer would be able to opt-in to advertisements using a familiar, standardized API. The creation of an `ADVERTISING` permission also allows the Android Market to become advertising-aware. The Market could allow users to sort applications based on whether they include advertisements. Additionally, the Market could support a feature that allows users to switch to the paid version of an application (e.g., if the user does not like the advertisements in the free version).

Section 4 discusses the AdDroid implementation in more depth. Section 5.6 discusses AdDroid's ability to support multiple advertising networks further.

3.3.1 Existing Applications and Libraries

AdDroid does not prevent the continued use of the existing bundled library model. Applications and advertising networks that do not use AdDroid simply do not receive the benefits of the system. Existing applications and advertising networks will continue to function normally. In the long term, we would expect the benefits of AdDroid

and changing user expectations to entice developers to migrate their applications and advertising libraries.

3.3.2 Limitations

The AdDroid proposal has two drawbacks. First, the Android development team would need to implement AdDroid and incorporate the functionality of existing advertising libraries into the Android API. However, we show in Section 5.6 that advertising functionality is relatively simple to support. Second, existing advertising networks are currently able to declare and update their APIs at their own pace, independent from Android release cycles. Under AdDroid, advertising networks would be constrained by the advertising support in the Android SDK. For small configuration changes, such as adding new advertising networks to AdDroid, the Android Market could disseminate signed configuration update files that describe advertising networks' web service APIs. However, larger changes to advertising networks might require platform updates. Since AdDroid does not preclude the use of the existing model, advertisers or developers unhappy with these limitations can simply not use the system. The AdDroid solution is less agile than traditional models, but we believe the security gains and ecosystem advantages outweigh the loss of flexibility.

4. IMPLEMENTATION

We implemented our AdDroid proposal to illustrate its details and demonstrate its feasibility. AdDroid consists of three parts: a userspace library that is part of the Android SDK, a new Android system service, and Android permissions. Our proof-of-concept implementation of the AdDroid prototype integrates these three components into the Android Open Source Project, version 2.3.3 (Gingerbread).

4.1 AdDroid Library API

The AdDroid userspace library provides developers with a public API, i.e., the classes and methods that developers invoke when writing applications. It supports the insertion of advertisements into applications' user interfaces and relays data between the application and the AdDroid system service. The library includes a new user interface element to display advertisements (an "AdView"), callback listeners, and support for a wide range of user tracking data.[3]

After surveying various advertising networks, we based the design of the API on the AdMob advertising SDK. AdMob is well-designed and supports most of the features of other advertising networks. Starting with the AdMob API, we made changes to generalize the API and extend it to other advertising networks. The AdDroid library allows developers to specify which advertising networks they would like to use, and allows use of multiple advertising networks in one application. A separate advertising network can be specified for each AdView, providing great flexibility. The AdDroid API is the same for all applications, regardless of which advertising network they use. Appendix A contains detailed description of the AdDroid API and sample usage.

[3]Applications often ask users for information such as their gender, which they then share with advertising networks. This data is distinct from the user data stored by the operating system.

Since the AdDroid library exists in userspace, it runs with the host application's permissions. We do not make any assumptions about what permissions the host application has. Furthermore, a grayware application could arbitrarily tamper with the userspace library. Consequently, the AdDroid library does not perform any privileged operations. Whenever an application requests a new advertisement, the library makes a `fetchAd` IPC call to the AdDroid system service which in turn performs the necessary privileged operations. Although the AdDroid userspace library does not perform any privileged operations, it contains the majority of the advertising functionality.

When a user clicks on an advertisement an `OnClickListener` is invoked, spawning a web browser directed to the advertisement's target URL. This action is how advertising services and applications track advertisement clicks. Launching a web browser is only one of several possible desired actions. Other behaviors can be added as needed by advertisers.

4.2 AdDroid System Service

The AdDroid system service runs with other Android system services as part of the Android system service process. This service's only job is to receive advertising requests from applications via the AdDroid userspace library and return advertisements. When the AdDroid system service receives an advertisement request, it establishes a network connection to the appropriate advertising network, transmits data to the advertising network, and stores the resulting advertisement. The AdDroid library then makes a follow-up IPC call to the AdDroid system service to retrieve the advertisement.

The data sent to the advertising network during the transaction might include configuration information (e.g., the application's customer number or the size of the requested advertisement), tracking data collected by the application, or advertising context-specific information specified by the application. If the application has the `LOCATION_ADVERTISING` permission, then the advertising network will also receive the phone's current location. Some advertising networks also request the phone's unique identifier. Our current prototype sends the telephony ID (i.e., IMEI, MEID, or ESN), but a full implementation could supply advertising networks with an alternate identifier. Alternatives include the `ANDROID_ID` (a unique ID that is not tied to the carrier), the `android.os.Build.SERIAL` field, or another identifier that is application-specific rather than device-wide.

4.3 Android Permission Changes

Applications should only be able to fetch ads through the AdDroid system service if they have the `ADVERTISING` or `LOCATION_ADVERTISING` permissions. To enforce this restriction, the AdDroid service verifies its caller's permissions. The ability to verify caller permissions from within Android services is part of the core Android security model, which we leverage.

We protect the `fetchAd` IPC call with our new advertising permissions. The `ADVERTISING` permission gives applications the ability to call `fetchAds` and request generic advertisements based on data supplied by the application. When applications use `LOCATION_ADVERTISING` in addition to `ADVERTISING`, applications may request that `fetchAds`

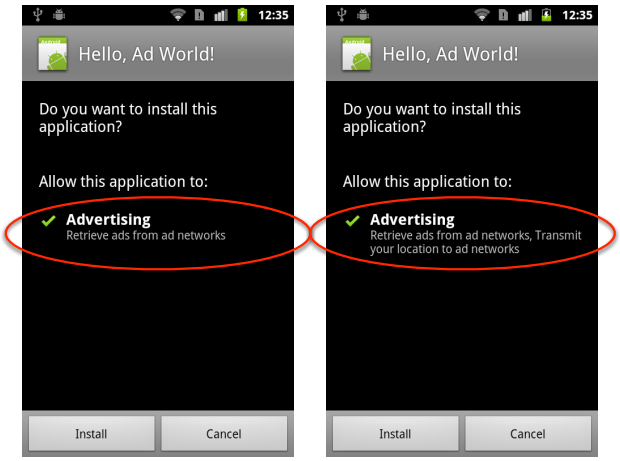

Figure 4: Installation screens of two applications requesting the new AdDroid permissions. The text of these permissions is for illustrative purposes only. Designing effective warnings is an open research problem [12].

sends location information to the advertisers as well. Figure 4 shows how the permissions appear to users during installation of our prototype.[4] Screenshots are provided for illustrative purposes; UI design is beyond the scope of this paper. Designing effective permission warnings that correctly convey security information to users is a problem in itself [12]. We leave it as an interesting open problem for usability experts how best to communicate this information to users.

4.4 Code Size

We found that AdDroid required relatively few modifications to the existing Android Open Source Project, version 2.3.3. The AdDroid userspace library required 560 new source lines of code including blank lines and comments (SLoC), and the new Android system service required 255 SLoC. Additionally, we modified or added 66 SLoC to the existing system to add support for the framework, system service, and new permissions. We flashed and tested the system on a GSM Samsung Nexus S unlocked consumer phone.

5. MEASUREMENT STUDY AND ADDROID EVALUATION

We analyzed a set of 964 Android applications to evaluate the real-world impact of AdDroid on applications and advertising. We performed a static permission analysis on the applications to determine the permissions used by the host applications and their libraries; Section 5.1 explains our methodology and Section 5.2 describes our data set. We then present results on how often applications request permissions solely for advertising (Section 5.3) and how many advertising libraries have access to dangerous permissions that are not necessary for advertising (Section 5.4). AdDroid would remove all of the overprivilege identified by this measurement study. In Section 5.5 we investigate the permissions commonly used by advertisers. In Section 5.6, we discuss how difficult it would be for developers to switch to AdDroid.

[4]These screenshots show the installation screen for *manual* installation, rather than for an installation from the Android Market. This is because we did not want to place an application in the Android Market solely for these screenshots.

5.1 Methodology

We used Stowaway [9] to perform a permission analysis of 964 real-world applications and their advertising libraries. Our set of applications consisted of the 764 most popular free applications, the 100 most popular paid applications, and the 100 most recently added or updated free applications from the official Android Market in October 2010.

The first step of our measurement study was to split each application into two parts: the application itself, and any advertising libraries packaged with the application. We disassembled applications using Dedexer [5] and then used namespaces to separate advertising-related code and XML files from the main application. For example, code in the `com.admob.android.ads` namespace belongs to the AdMob advertising library. We focused our study on applications that contained advertising libraries. Next, we ran Stowaway on the applications and their affiliated advertising libraries, which resulted in a list of permissions used by each application and its advertising libraries.

Stowaway's permission analysis can be imprecise. Dead code (i.e., code that will not run during normal execution) can cause Stowaway to falsely report that an application uses a permission, or Java reflection can cause it to falsely report that an application does not use a permission. To validate Stowaway's results, we manually reviewed 25 applications' use of the `INTERNET` permission. We randomly selected 25 applications from the set of top free applications with advertising libraries. We manually exercised their user interfaces in an emulator while recording all network communication with Wireshark. Using the Wireshark traces, we determined whether the applications use the Internet to communicate with advertisers, other parties, or both.

5.2 Data Set Characteristics

473 of the 964 applications (49%) contain libraries from at least one advertising network. We found advertising libraries from nine advertising networks in the 473 applications: AdMob, AdSense, AdWhirl, Flurry, Medialets, Millennial Media, MobClix, Quattro Wireless, and ZestAdz. Our data set includes multiple versions of each advertising library because developers do not always update their applications to include the latest versions. Some applications use multiple advertising libraries. Table 1 shows the popularity of each advertising network and the distribution of advertisements across free and paid applications. Unsurprisingly, free applications are more likely to contain advertising libraries: 16% of the top paid applications include advertising libraries, compared to 50% of the top free applications and 72% of the recently-uploaded free applications.

5.3 Application Overprivilege

We identified advertising-related overprivilege in the 473 applications with advertising libraries. An application is

	AdMob	AdSense	AdWhirl	Flurry	Media-lets	Millen-nial	MobClix	Quattro	ZestAdz	*Total with ads*	*Total apps*
Top Free	231	155	44	79	6	32	34	38	4	**385**	**764**
Top Paid	4	6	0	11	0	0	2	1	0	**16**	**100**
Recent Free	63	10	2	2	0	1	0	1	0	**72**	**100**

Table 1: The prevalence of advertising libraries among 964 applications. Some contain multiple advertising libraries.

Permission	Requested only for ads	Total apps requesting
`INTERNET`	131	809
`LOCATION`	121	361
`READ_PHONE_STATE`	40	312
`ACCESS_NETWORK_STATE`	36	395

Table 2: The number of applications that request a permission exclusively for the purpose of advertising. For comparison, we also show the total number of requests for that permission in the set of 964 applications.

overprivileged by advertising if:

- The application requests a permission.
- The application itself does not use the permission.
- One of its advertising libraries uses the permission.

We focus on overprivilege pertaining to the four permissions related to common advertising library operations. These four permissions control access to the Internet, the phone's location,[5] the state and identity of the phone, and the state of the phone's network connectivity.

We find that advertising libraries overprivilege 218 of the applications (46% of the applications with advertisements, and 23% of all 964 applications). 110 applications request multiple permissions that are only used for advertising. Table 2 shows how often each of the four advertising-related permissions contribute to overprivilege. Notably, more than a third of all applications that request access to the user's location do so *only* for advertising. 56% of ad-supported applications (34% of all applications) that request a location permission use it only for advertising. With AdDroid, requests for advertising permissions would replace all of these unnecessary permission requests.

We performed a manual review to validate the results of the automated analysis. The manual review found that 9 of 25 randomly-selected applications (36%) only use the `INTERNET` permission for advertising. Stowaway identified 6 of the 25 applications as overprivileged. We determined that the discrepancy was due to dead code: 3 of the applications contain dead code that would access the Internet if it were reachable. For example, one application contains unfinished code for connecting to Facebook. However, the code is not currently connected to any user interface elements (e.g., menu items or buttons). We did not observe any instances in which Java reflection caused Stowaway to falsely state that an application does not use a permission.

[5]There are technically two permissions that control location: one for GPS location, and one for cellular network location. We group the two location permissions together for this discussion.

5.4 Advertising Library Overprivilege

Currently, a malicious or aggressive advertising library has access to all of its host application's permissions. In the set of 473 applications with advertising libraries, 398 applications (84%) request permissions beyond those needed by the advertising library. For example, 149 applications' advertising libraries inherit the ability to read the user's contacts, which a malicious advertising network could use for aggressive marketing campaigns. With AdDroid, advertising libraries would not have access to these extra permissions: AdDroid prevents advertising networks from abusing permissions that applications have incidentally requested.

5.5 Advertising Functionality

AdDroid's system service is an advertising service that supports Internet advertisements with differing behavior based on phone identity, and user's location. We evaluate whether this is sufficient to support the same functionality as the advertising libraries in our data set.

The permissions used by the AdDroid system service satisfy the functionality of 456 of the 473 applications' advertising libraries. However, we find that 17 applications include advertising libraries that use two additional, unsupported permissions:

- `GET_TASKS`. Some versions of MobClix use this permission to check that the advertisement is currently displayed in the foreground. While in principle this permission could be used to spy on users' application preferences, we did not observe MobClix using it in this fashion. Twelve applications include the version of MobClix that uses this permission.

- `VIBRATE`. Some versions of Medialets support advertisements that vibrate the phone to catch the user's attention. Five applications use a version of Medialets that includes this functionality.

The advertising libraries that use these permissions are not fully compatible with the AdDroid proposal.

AdDroid could support the advertising-related behavior associated with the `GET_TASKS` permission by adding an API call that allows an advertising service to determine whether it is in the foreground, without providing it with the full list of applications. This would allow AdDroid to support MobClix's desire to restrict certain actions to the foreground. We have not yet investigated whether there are technical challenges for providing this feature. We do not recommend adding support for vibrating the phone; we feel that these are intrusive advertising practices. As such, AdDroid does not support all of Medialets' functionality, but we believe this benefits users.

5.6 API For Developers

For AdDroid to be successful, developers must be able to use its API. We have two usability goals: (1) a single API able to support multiple advertising networks, and (2) developers need to be able to easily port their applications from existing libraries to the new AdDroid API. We evaluate this with our prototype API and two sample applications.

Towards these goals, our proof-of-concept implementation of AdDroid (Section 4) supports the core advertising and user interface interactions of the AdMob and Millennial Media advertising networks. We do not currently support advanced functionality such as interstitial or video advertisements, but this is not an inherent limitation. AdMob, Millennial Media, and Google AdSense make up 69% of ad network usage in our dataset (Table 1). We support AdMob and Millennial Media, and Google has since deprecated the AdSense SDK in favor of AdMob. Consequently, the AdDroid API already covers 69% of the market. Most other advertising networks are similar to AdMob and Millennial Media, and we believe that their needs could also be met with only small changes to our implementation.

We built two sample applications that use the AdDroid API to load real advertisements from advertising networks. This required a modification to our implementation because our proof-of-concept implementation of AdDroid interacts with our own advertising server rather than actual advertising networks for ethical reasons. Existing advertising libraries are not open source, and reverse engineering an undocumented protocol could lead to excessive charges to advertisers for impressions or clicks that did not occur. Instead, for this demonstration only, we replaced the AdDroid system service with existing advertising libraries. The sample applications invoke API calls in the AdDroid library, which proxies the calls to an advertising library, which in turn communicates with the advertising network server.

The first sample application uses the AdDroid API to fetch AdMob advertisements. The second uses the AdDroid API to fetch Millennial Media advertisements. We bundled them with the AdDroid userspace library and the original advertising libraries. The applications use the AdDroid API to fetch advertisements and do not directly invoke the original advertising libraries. Both of the sample applications are able to correctly display advertisements from these advertising networks through the AdDroid API, and their use of the AdDroid API is nearly identical to how the original advertising APIs are used. Switching from the original library to the AdDroid API required us to modify less than 20 lines of code in our sample application.

6. RELATED WORK

Concurrent with this work, several other researchers have looked at the problem of Android overprivileging and advertising. AdSplit [25] approaches the problem of advertising overprivileging by running advertising libraries as separate applications, and uses prior work [6] to allow both applications to share the screen. AdSplit also performs a measurement study similar to our own, yielding similar overprivileging results. Leontiadis et al. [17] also use separate applications to limit advertising library overprivilege, leveraging in-application widgets and IPC instead of screen sharing. These works are similar to the proposed approach in Section 3.2. Each work encounters technical hurdles and limitations similar to those outlined in our proposed approach. Some of these limitations are solved with added complexity to the Android platform, and others are left as open problems. AdSplit also speculates that its ideal deployment is as part of the core Android distribution, similar to our approach with AdDroid.

Prior work on Android overprivilege has not considered the impact of advertising. Recent studies [11, 9] have explored whether Android applications request more permissions than they require to function. However, they considered advertising libraries to be a legitimate use of permissions, whereas we do not. Our measurement study focuses specifically on overprivilege that stems from advertising.

Other researchers have explored information leakage to advertisers. Grace et al. [13] used static analysis to perform a measurement study of the type and quantity of private information sent to advertisers across over $100,000$ applications. They found that advertising networks were opportunistically checking for and making use of host application permissions. Grace et al. also found that some advertising networks were downloading code over HTTP and executing that code during runtime. AdDroid alleviates these problems by isolating the privileged parts of advertising libraries from the host application, and ensuring that such privileged code is secure. TaintDroid [8] and PiOS [7] study information flow on the two most prevalent smartphone platforms: Google's Android and Apple's iOS. They discovered that smartphone applications often divulge sensitive personal information (e.g., unique device identifiers, location) to application developers and other third parties. Known advertising networks are among the list of third-party recipients of data. Journalists [26] conducted their own study of Android and iPhone applications with similar results. AppFence [15] extends TaintDroid to track more types of user data and detect exfiltration. The user is warned at the point of exfiltration and can disallow the operation. TaintDroid and AppFence are intended to detect and warn about the exfiltration of data by any party. In contrast, AdDroid prevents advertising networks from accessing information beyond location and device identifiers without user involvement. AdDroid simply disallows invasive data collection by advertisers; unlike approaches that rely on taint tracking, AdDroid does not incur any additional overhead at runtime.

AdJail [18] is a mechanism to protect desktop browsers from malicious advertisements. AdJail uses policies and sandboxing mechanisms to isolate users from advertisements and prevent the disclosure of sensitive user data. The authors intend AdJail for web site publishers who want to protect their visitors, much like we direct AdDroid's functionality to application developers to protect their customers.

Quire [6] is a proposed solution to the problem of provenance on Android. Currently, the identity of the original caller in Android IPC can quickly become obfuscated. Quire approaches this problem with call chains and message signing. The authors proposed that Quire could be used to help build a new advertising ecosystem with separate user-level advertising applications. Quire's objective, with respect to advertising, is to prevent advertising fraud by certifying that applications properly display and users

click advertisements. Our objective with AdDroid is to enhance privacy by preventing advertising libraries from disclosing private data.

Privacy on mobile platforms is a growing public and legal concern. MobiAd [14] is a proposed solution to allow for targeted advertising while maintaining user privacy and anonymity. The authors also note concerns over the trade-off users make for these interest- and location-based advertisements. They conclude by stating that there is insufficient data to know when and where users are willing to accept advertisements. Cleff [3] argues that legislation is slow and well behind technological advancements and discusses the legal problems this raises in regards to protecting users' privacy. Cleff asserts that users need control over their private data and the types of advertising they receive.

7. CONCLUSION

With the widespread use of advertising in Android applications, it is important to examine the security and privacy implications of the existing advertising model. Users of advertisement-supported applications can fall victim to advertising networks tracking their behavior without their knowledge, grayware, buggy-but-benign applications, and malicious or vulnerable advertising networks.

To combat the privacy and security threats related to advertising, we propose *AdDroid*, an advertising API integrated into the Android platform. AdDroid uses privilege separation to isolate privacy-sensitive advertising network functionality from applications and introduces new advertising permissions to protect the API. We have a proof-of-concept implementation of AdDroid and its advertising API. Through the use of AdDroid we believe the Android ecosystem can evolve and directly integrate advertisements into the fabric of the market. Such integration can provide not just increased user privacy and security, but also economic benefits and incentives for advertisers, developers, and the platform itself.

To demonstrate the need for AdDroid, we performed a study of the Android Market and found that 46% of advertisement-supported applications suffer from overprivilege resulting from advertising library use. We also find that 34% of all applications (56% of advertisement-supported applications) request access to location information solely for advertising. Use of AdDroid would prevent this overprivileging due to advertising. Instead, applications would ask for the `ADVERTISING` or `LOCATION_ADVERTISING` permissions, and advertising networks would not gain access to applications' other permissions.

8. ACKNOWLEDGEMENTS

We would like to thank Úlfar Erlingsson for this helpful feedback throughout the project. This research was supported by Intel through the Intel Science and Technology Center for Secure Computing, by the Office of Naval Research under MURI Grant No. N000140911081, by a Facebook Fellowship, and by a gift from Google.

9. REFERENCES

[1] AdMob: Mobile Advertising. `http://www.admob.com/`.

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

[3] Cleff, E. B. Privacy Issues in Mobile Advertising. *Int. Review of Law, Computers & Technology* (2007).

[4] 2009 Study: Consumer Attitudes About Behavioral Targeting. `http://www.truste.com/pdf/TRUSTe_TNS_2009_BT_Study_Summary.pdf`, 2009.

[5] Dedexer User's Manual. `http://dedexer.sourceforge.net/`.

[6] Dietz, M., Shekhar, S., Pisetsky, Y., Shu, A., and Wallach, D. S. Quire: Lightweight Provenance for Smart Phone Operating Systems. In *USENIX Security Symposium (SSYM)* (2011).

[7] Egele, M., Kruegel, C., Kirda, E., and Vigna, G. PiOS: Detecting Privacy Leaks in iOS Applications. In *Network and Distributed System Security Symposium (NDSS)* (2011).

[8] Enck, W., Gilbert, P., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., and Sheth, A. N. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (2010).

[9] Felt, A. P., Chin, E., Hanna, S., Song, D., and Wagner, D. Android Permissions Demystified. In *ACM Conference on Computer and Communication Security (CCS)* (2011).

[10] Felt, A. P., Finifter, M., Chin, E., Hanna, S., and Wagner, D. A Survey of Mobile Malware in the Wild. In *ACM Workshop on Security and Privacy in Mobile Devices (SPSM)* (2011).

[11] Felt, A. P., Greenwood, K., and Wagner, D. The Effectiveness of Application Permissions. In *USENIX Conference on Web Application Development (WebApps)* (2011).

[12] Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. Android Permissions: User Attention, Comprehension, and Behavior. Tech. Rep. UCB/EECS-2012-26, University of California Berkeley, 2012.

[13] Grace, M., Zhou, W., Jiang, X., and Sadeghi, A.-R. Unsafe Exposure Analysis of Mobile In-App Advertisements. In *Conference on Security and Privacy in Wireless and Mobile Networks (WiSEC)* (2012).

[14] Haddadi, H., Hui, P., Henderson, T., and Brown, I. Targeted Advertising on the Handset: Privacy and Security Challenges. In *Pervasive Advertising*. 2011.

[15] Hornyack, P., Han, S., Jung, J., Schechter, S., and Wetherall, D. These Aren't the Droids You're Looking For: Retrofitting Android to Protect Data from Imperious Applications. In *ACM Conference on Computer and Communication Security (CCS)* (2011).

[16] Kelley, P. G., Benisch, M., Cranor, L., and Sadeh, N. When Are Users Comfortable Sharing Their Locations With Advertisers? In *Conference on Human Factors in Computing (CHI)* (2011).

[17] Leontiadis, I., Efstratiou, C., Picone, M., and Mascolo, C. Don't kill my ads!: Balancing Privacy in an Ad-Supported Mobile Application Market. In *Workshop on Mobile Computing Systems & Applications (HotMobile)* (2012).

[18] Louw, M. T., Ganesh, K. T., and Venkatakrishnan, V. AdJail: Practical Enforcement of Confidentiality and Integrity Policies on Web Advertisements. In *USENIX Security Symposium (SSYM)* (2010).

[19] McDonald, A. M., and Cranor, L. F. Americans' Attitudes About Internet Behavioral Advertising Practices. In *Workshop on Privacy in the Electronic Society (WPES)* (2010).

[20] Miller, B., Pearce, P., Grier, C., Kreibich, C., and Paxson, V. What's Clicking What? Techniques and Innovations of Today's Clickbots. In *Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)* (2011).

[21] Mobile Advertising with Millennial Media. `http://www.millennialmedia.com/`.

[22] Pathak, A., Charlie, Y., and Zhang, H. M. Where is the energy spent inside my app? Fine Grained Energy Accounting on Smartphones with Eprof. In *European Conference on Computer Systems (EuroSys)* (2012).

[23] Provos, N., Friedl, M., and Honeyman, P. Preventing Privilege Escalation. In *USENIX Security Symposium (SSYM)* (2003).

[24] Saltzer, J. H., and Schroeder, M. D. The Protection of Information in Computer Systems. In *IEEE* (1975).

[25] Shekhar, S., Dietz, M., and Wallach, D. S. AdSplit: Separating Smartphone Advertising From Applications. *CoRR abs/1202.4030* (2012).

[26] Thurm, S., and Kane, Y. I. Your Apps Are Watching You. `http://online.wsj.com/article/SB10001424052748704694004576020083703574602.html`, Dec. 2010.

APPENDIX

A. SAMPLE API AND USAGE

Listing 1 shows our AdDroid API, abbreviated for space. Listing 2 shows sample usage of the AdDroid API by the demo application called AdDroidDemo. This code demonstrates how an AdDroid ad can be instantiated.

```
Interface AdListener
  void onCacheReceive(Ad ad)
  void onDismissScreen(Ad ad)
  void onFailedToRecieveAd(Ad ad,
         AdRequest.ErrorCode error)
  void onForcedPresentScreen(Ad ad)
  void onLeaveApplication(Ad ad)
  void onPresentScreen(Ad ad)
  void onReceiveAd(Ad ad)

Class AdNetwork
  public AdNetwork(AdNetwork.NetworkType network,
           java.lang.String applicationId)
  public java.lang.String getId()
  public AdNetwork.NetworkType getNetwork()

static enum AdNetwork.NetworkType
  ADMOB  MILLMEDIA  MOBCLIX
```

```
Class AdProperties
  public AdProperties(int width, int height,
           AdProperties.Align alignment,
           int refresh)
  public AdProperties.Align getAlignment()
  public java.lang.String toString()
  public static AdProperties BANNER
  public static AdProperties BANNER_BOTTOM
  public static AdProperties BANNER_TOP
  public static AdProperties IAB_BANNER
  public static AdProperties IAB_LEADERBOARD
  public static AdProperties IAB_MRECT
  public static int NO_REFRESH

static enum AdProperties.Align
  BOTTOM  TOP  UNSET

Class AdRequest
  public AdRequest()
  public void addExtra(String key, Object value)
  public void addKeyword(String keyword)
  public Map<String, Object> getRequestMap()
  public void setBirthday(String birthday)
  public void setExtras(Map<String,Object> extrs)
  public void setGender(AdRequest.Gender gender)
  public void setKeywords(Set<String> keywords)
  public void setLocation(boolean location)
  public void setTesting(boolean testing)

static enum AdRequest.ErrorCode
  INTERNAL_ERROR  INVALID_REQUEST  NETWORK_ERROR
  NO_FILL  UNKNOWN

Class AdView extends RelativeLayout implements Ad
  public AdView(Activity activity,
           AdProperties props, AdNetwork network)
  public boolean isReady()
  public boolean isRefreshing()
  public void loadAd(AdRequest request)
  public void setAdListener(AdListener listener)
  public boolean stopLoading()
```

Listing 1: Our abbreviated AdDroid API as found in package com.ads.addroid.

```
public class AdDroidDemo extends Activity {
  // Called when the activity starts or restarts
  @Override
  public void onCreate(Bundle savedInstanceSt){
    super.onCreate(savedInstanceSt);
    setContentView(R.layout.main);
    // Specify the add type
    // (e.g., banner, interstitial, video)
    AdProperties props = AdProperties.BANNER;
    // Specify the ad network (e.g., AdMob)
    // and the associated unique ID
    AdNetwork network =
      new AdNetwork(AdNetwork.NetworkType.ADMOB,
                    "ADVERTISER_ID");
    // Instantiate the view for the ad
    AdView adView =
      new AdView(this, props, network);
    // Lookup your LinearLayout assuming its
    // been given the id "main_layout"
    LinearLayout layout =
      findViewById(R.id.main_layout);
    // Add the AdView to the layout
    layout.addView(adView);
    // Instantiate the ad request
    AdRequest adReq = new AdRequest();
    // Request and load the ad
    adView.loadAd(adReq);
  }
}
```

Listing 2: A complete AdDroidDemo activity showing sample usage of the AdDroid API.