SPrivacy: Context dependent user data privacy in Android

Bin Liu*, Prajit Kumar Das†, Anupam Joshi†, Tim Finin† and Hongxia Jin*
*Samsung Research America
Email: bin2.liu,hongxia.jin@samsung.com
†University of Maryland Baltimore County
Email: prajit1,joshi,finin@umbc.edu

***Abstract*—Mobile devices that we use, have a lot of in-built sensors that are capable of collecting data about the user. We also use our devices to access our contacts, photos, calendar, email, social networking sites and other internet or ad-hoc network based resources. As a result, a substantial amount of our personal information either is accessed and stored on these devices. Mobile operating systems expose APIs to applications(or apps) installed on the phone to access these stored data. The intention is to provide services to the user of the mobile device. Unfortunately, the current privacy and security mechanisms that control the data access use an "all or nothing" policy. Such a policy is inadequate in controlling data in specific contextual situations. There has been a lot of research in this area ranging from detection of data flow to prevention of data flow. These solutions achieve the required control by modifying the APIs or the mobile operating system itself. In this paper we present SPrivacy, a system built which uses privacy policies that define which app will be able to get access to the data, to what degree will this access be allowed and under what user context will such an access be allowed. We use SPrivacy to control "suspect" apps from accessing private information on the user's mobile device given the user context. We do this by static analysis and "constants" manipulation in the app's installer file. We make no changes to the mobile device's operating system or its APIs. We have implemented SPrivacy as per the definitions of the XACML standard for Attribute Based Access Control. We evaluate SPrivacy to show that it causes minimal overhead on the performance of the system.**

I. Introduction

Arguably one of the most popular mobile operating systems, by number of devices in the market today, is Google's Android [1]. Our mobile devices, which use the Android operating system, are increasingly being used to access and store our personal data like our contacts, email, calendar data, photos, chats, social networking data, financial data, as well as contextual data like location and activity information. At present, the security and privacy of users' data on these devices is controlled by a static install time permission acquisition process. The permissions requested are of the "all or nothing" form meaning either the user grants all the permissions or else they cannot use the app in question.

One fundamental issue with the permission model for Android apps is that the user does not get to decide, if they want to protect certain specific contents on their phones. Let's consider the following use case: John Doe is at a party. He wants to keep the pictures taken during the party as private. He had previously installed an app "InstaUpload" that automatically synchronizes all his gallery data to an online account and tweets about the photos. John usually wants to avail the services of the app but in this specific contextual scenario he prefers that the app should not upload the pictures. The app obviously has the permission required to access the photos on John's mobile so there is no way for John to stop it.

It might sometimes be important to the user of a mobile device to be able to protect certain content on their phones. huge Empowered with these permissions, apps are then requesting this user data via various APIs in the Android framework. The two main mechanisms for apps to access data on Android include, using an Intent or using an explicit call to a ContentProvider. Given the highly personal nature of the data it is imperative that we protect it from being used by "rogue" apps with malicious intent. However, It might be the case that a user wants to use a particular app because they wanted to test out certain features provided by it.

c and iOS. A simple review of the literature and online documentation for Android tells us that iOS security is centralized in the settings of the device. Access to certain privileged data is controlled by first usage permission request basis. and Android security is follows an individual app permission model. Android Privacy is a big challenge. This paper is our attempt at solving it. The apps that we control are processed by a decompilation mechanism followed by static modification of constants in order to control access to Content Providers they use.

Broadly these will be the sections in the paper but we might change this later:

1) Introduction
2) Overview
3) System Design and Architecture
4) Implementation Details
5) Experimental Evaluations
6) Discussion and Related Work
7) Conclusion

II. Overview

...

Android mobile devices use Content Providers in order to allow applications (or apps in short), installed on the device,

access to user data. Android's permission mechanism controls whether such an access will be allowed or not. However, the current permission model for Android is a "all or nothing" model. The user has to either accept all the permissions that the app requests or they cannot install the app. Such a model is overtly restrictive and does not allow the possibility of execution time permission granting. It also does not allow the user to restrict the data or allow data access based on contextual situations. Current generation of mobile devices are capable of collecting a lot of user information using in-built sensors or by accessing user personal contacts, calendar, emails and messages. In such a situation it has become necessary that user have more control over their data. There needs to be a better way of controlling such data. In this paper we implement a new privacy aware middle-ware, SPrivacy. SPrivacy allows the users to dynamically control the data that an app is allowed to access.

Traditional solution(s) for controlling user data privacy on mobile devices have focused on two techniques:

1) Make changes to the mobile operating system and control how data flows from the creator to the consumer of the data on the device. This is done by creating custom ROMs or mobile operating systems and modifying the APIs in the custom operating system to achieve the goal mentioned before.
2) Obtain elevated privileges from the operating system in order to control how the data flows. In this mechanism an app is installed on the device with "root" privileges on the device so that the app is able to modify the behavior operating system APIs as per its needs.

We argue that both these techniques potentially create loopholes which might be used by a malicious app towards its own untoward actions. Our proposed mechanism allows us to have no changes on the operating systems but at the same time allows us to control the data that an app can access on the device. We achieve this by implementing SPrivacy which is a middle-ware that is capable, based on a list of settings and a URI redirection mechanism, of controlling the data app(s) get. In our mechanism we take apps from the Android app market and modify the Content Provider URIs they use in order to ensure they call our middle-ware instead. Our middle-ware on then determines if the app will be given access to the data or not.

III. System Design and Architecture

system design

IV. Implementation Details

...

Taming Information-Stealing Smartphone Applications (on Android) - Our closest competitor. However, they use only 24 apps to analyze their data. Which is odd and they use 13 "known to be rogue apps" based on TaintDroid system's analysis. The rogue apps leak information which are sensitive like IMEI number and location information. They do not expalin why they just use 24 apps though.

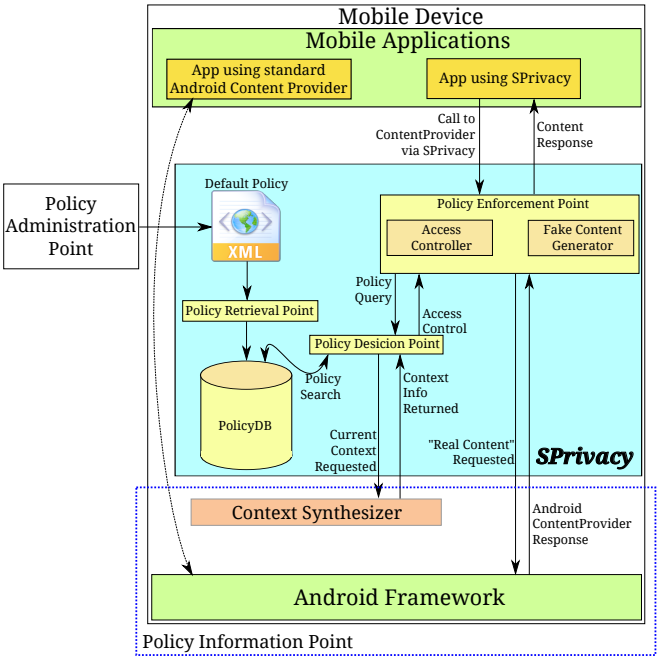

Fig. 1. High-level architecture of our system.

V. Experimental Evaluations

...

Taming Information-Stealing Smartphone Applications (on Android) - Our closest competitor. However, they use only 24 apps to analyze their data. Which is odd and they use 13 "known to be rogue apps" based on TaintDroid system's analysis. The rogue apps leak information which are sensitive like IMEI number and location information. They do not expalin why they just use 24 apps though.

VI. Related Work

...

Taming Information-Stealing Smartphone Applications (on Android) - Our closest competitor. However, they use only 24 apps to analyze their data. Which is odd and they use 13 "known to be rogue apps" based on TaintDroid system's analysis. The rogue apps leak information which are sensitive like IMEI number and location information. They do not expalin why they just use 24 apps though.

VII. Conclusion

The conclusion goes here.

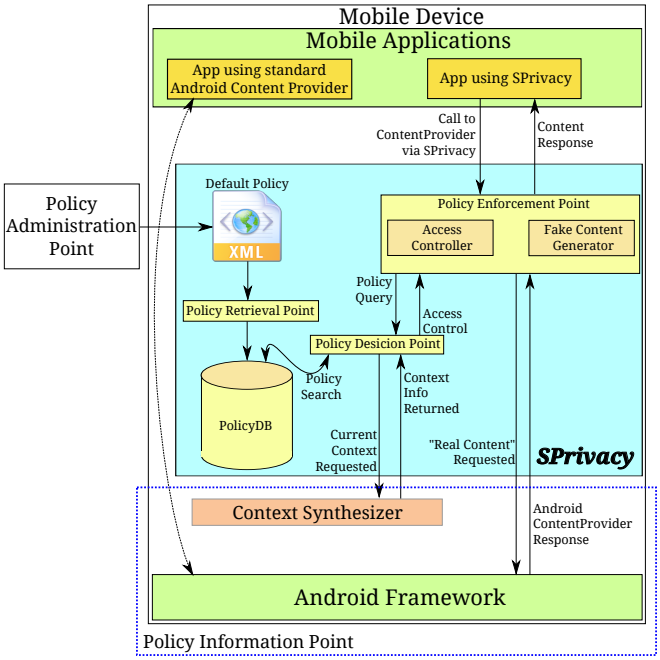

Fig. 2. High-level architecture of our system.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] H. McCracken, "Whos winning, ios or android? all the numbers, all in one place," April 2013. [Online]. Available: http://goo.gl/SteXKp