BFH1925019F

by Iftekhar Efat

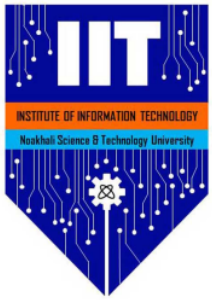
Submission date: 13-Mar-2022 01:57AM (UTC-0500)
Submission ID: 1782995741
File name: BFH1925019F.pdf (612.84K)
Word count: 2325
Character count: 12897

The policy of installing applications to the phones and tables

Ikra Chowdhury Nowkshi
BFH1925019F

March 13, 2022

Report submitted for **SE2206: Information Security** under BSc. in Software Engineering Program, Institute of Information Technology (IIT), Noakhali Science and Technology University



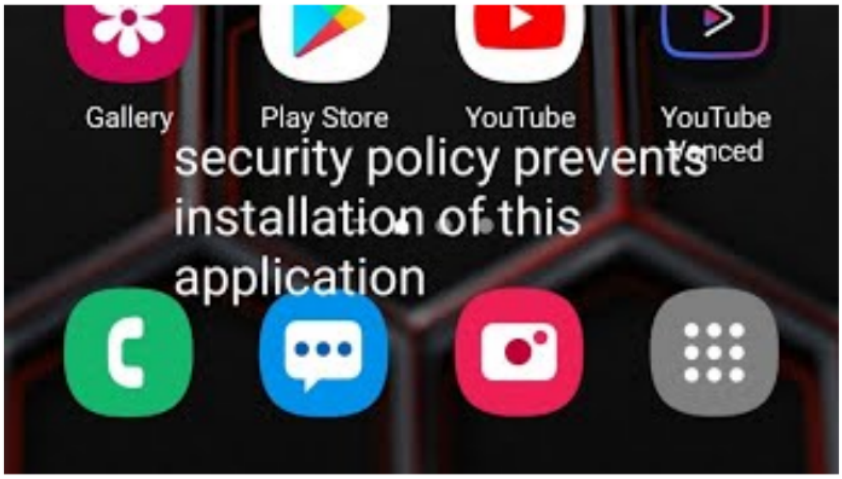
Project Area: **Information Security .....**
Project Supervisor: **MD. IFTEKHARUL ALAM EFAT**
Assistant Professor
Institute of Information Technology (IIT)
Noakhali Science and Technology University

Abstract

In cellular networks, phones with open operating systems like Google Android will soon be the norm. These systems expose previously unavailable phone and network resources to application developers. Increased exposure, on the other hand, comes with increased danger. Applications that are poorly or maliciously developed can compromise the phone and network. The goal of a Software Installation Policy is to identify permitted standard software titles, clearly communicate what is prohibited, and outline proper procedures for installation.The purpose of this report is to explore application installation policies on phone and tablets with the goal to understand security policy and its importance along exploring the present drawback.

1 Introduction

Mobile phones used to be simple gadgets that could only do basic phone operations. However, third-party developers are often responsible for additional programs such as games, productivity, and communication apps. The development of third-party apps has become a critical aspect in determining a platform's financial success. This has prompted major smartphone OS makers to give open development tools, including as APIs, emulators, and tools for building apps, even if the platform isn't totally open. Thousands of apps are currently accessible on the leading smartphone platforms, which users can usually install with a few key strokes through an on-device shop. Allowing customers to easily install a large number of apps from a range of developers poses security concerns. Users must decide whether or not they



can trust an app and its developer, as well as whether or not the app will break any other apps they have loaded. As a result, it is necessary to setup the platform in such a way that users are unable to install programs on their own. A solid software installation policy might help you avoid problems like these.

Software licensing:- issues One of the primary reasons why it is important to prevent users from installing software on their phones

Malicious software -Another reason for not allowing users to install applications of their own is because of the threat of malicious software. There are two different types of threats that you have to worry about when it comes to malicious software. One threat is malicious software that is piggybacked onto legitimate software.The other type of malicious software that we have to worry about is the type that gets installed without the user's knowledge or consent.

Other vulnerabilities-One last reason why a good software installation policy is important is because unauthorized software can increase the chances of the system being exploited. There is a law of computing that states that the greater the amount of code that's executing on the system, the better the chances that the code will contain at least one critical security vulnerability.

Google's Android is the most popular and widely used smartphone operating system

today. To have a better grasp of the issue, we will first focus on the policy of Android application installation in this study. Due to its ubiquity (at the time of writing, the smartphone OS with the biggest market share) and open architecture, we conduct an in-depth analysis of the Android OS.

2 Android Security Model

The Android security model is based on the Application Sandbox. Android maintains sandbox separation, which prevents apps from affecting or interfering with one another. Each sandbox, which may contain one or more apps, is given a unique Linux user ID (UID) by Android (see Section 4). This method enables apps to have private file storage, memory, and process space, all of which are enforced by the kernel.If necessary, developers can include a list of permissions in the app's manifest to seek special privileges (e.g., use of the camera, GPS sensor, or microphone). At installation time, the manifest is reviewed, and the user is asked whether they want to accept or reject the permission set. Permissions cannot be rejected individually in standard Android builds. As a result, developers expect that after their app is loaded, users will be able to access all of the functionality protected by the rights granted.Code signing is used in conjunction with the app sandbox to provide several fundamental aspects of the Android security model. The developer's certificate is used to limit who can deliver software updates to the app as well as the app's interprocess communication (IPC) capabilities and if certain permissions can be gained.

For example the MASTERCLEAR permission allows an app to remove all user

data and restore the device to its factory state. While any app can seek the MASTERCLEAR permission, it can only be granted to apps signed with the same key as the system image. Signature or signatureOr System permissions are the names for certain permissions. Additionally, third-party developers can create public APIs and secure them using signature-level rights.

3 Android's vulnerabilities

Malicious apps and libraries that abuse their privileges or even employ root exploits to steal security and privacy critical information; taking advantage of exposed interfaces and files confused deputy attacks and collusion assaults have all been demonstrated to be vulnerable to Android.
[1] Android's open source nature, on the other hand, has made it particularly desirable to academic and industrial security research. Various extensions to Android's access control framework have been proposed to address specific problem sets such as user privacy protection, application centric security such as Saint, which enables developers to protect their application interfaces, establishing isolated domains (use of the phone in a private and corporate context), mitigation of collusion attacks, and extending Android's Linux kernel with Mandatory Access Control Analyzing the large body of literature on Android security and privacy one can make the following observations:To begin, practically all security enhancements to Android are mandatory access control (MAC) techniques suited to the specific semantics of the addressed problem, such as establishing fine-grained access control to user's private data or safeguarding platform integrity. Furthermore, these solutions are lacking in one essential aspect: protection mechanisms only work at a single system abstraction layer, i.e., either the middleware (and/or application) layer or the kernel-layer.As a result, they overlook the Android OS's unique design, which emphasizes the importance of each of its two software layers (middleware and kernel) inside its respective semantics for the desired overall security and privacy. Only a few solutions take into account both levels, but they only enable a fairly static policy and lack the flexibility needed to instantiate multiple security and privacy models.

Another problem is that determining what constitutes harmful behavior in an

Android application is difficult. An install-time application permission system controls access to privacy- and security-relevant aspects of Android's API. Users are notified about the data and resources that an application will have access to, and consent is necessary before the app can be installed. The application package de-

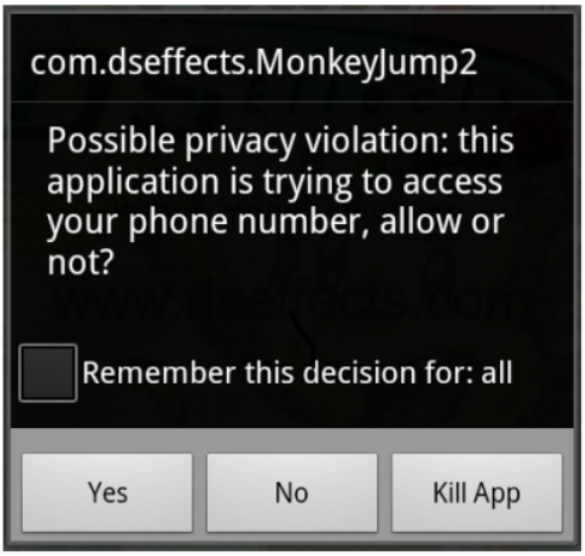
clares these explicit permissions. Install-time permissions give users more privacy control, although they are frequently coarse-grained. Permissions granted at the time of installation are valid for as long as the app is present on the device.While an application may properly request Internet access, it is unclear what connections it may make with potentially malicious distant servers. Similarly, an application may properly necessitate the transmission of SMS messages. According to a recent Android applications analysis [4], the mere request for SMS permission by an application can be deemed malicious, with 82 percent of malicious applications requiring permissions to access SMS. Once the SMS permission is granted, there are no checks to prevent the application from sending SMS messages to premium numbers without user consent.Many of the issues linked with application component interactions, delegation of permission, and permission escalation attacks are revealed in a recent survey [3] due to insufficient or missing security policy descriptions by developers. This inspired early work on an Android security policy extension.
Again an issue is that any strategy that relies on policy extensions to improve the platform's security and privacy controls so far necessitates significant changes to the operating system. This has serious usability concerns and will stymie any widespread adoption efforts. There are a variety of tablet and phone devices with various hardware configurations, each with its own Android OS version, customizations, and device drivers.This phenomena, commonly known as the infamous Android version fragmentation problem [2], indicates that customizing Android for all potential devices in the wild is challenging. It's considerably more difficult to ask a regular user to apply a security framework's source patch and compile the Android source tree for their own device.Many OS-based Android security projects will be unable to get widespread adoption due to these concerns. Alternatively, due to skewed incentives from various parties, it is as difficult to bring Google, phone manufacturers, and cellular providers together to offer security enhancements at the consumer market level.

4 Policies

The following subsections describe a set of policies which leverage the security policy in term of application installation.

4.1 Privacy Policy

The most obvious collection of policies that can be defined is those that pertain to the privacy of users. These policies safeguard the user's personal information, including their IMEI, IMSI, phone number, location, SMS messages, phone conversations, and contact list. These policies can be verified by monitoring access to the Framework APIs' system services. While there are numerous APIs for accessing system services, they all boil down to one call to the ioctl() system call. We can determine which service is being visited and inform the user by monitoring calls to ioctl() and analyzing the data transmitted in the call.



4.2 Network Policy

A set of network policies that limit how an application is authorized to interact with the network, similar to how we enforce privacy policies is applied. Because the Android permission mechanism provides for unfettered Internet access its does not give proper protection.This policy could be improved by the following:

- restrict the application to only a particular web domain or set of IP addresses
- restrict the application from connecting to a remote IP address known to be malicious

4.3 Privilege Escalation Policy

Certain sorts of escalation of privilege attacks can be avoided by knowing the attack signatures based on dubious executables.The warning indicates that the software is attempting to get root access through the (s)u command on a potentially rooted phone.The policy also includes mechanism which alerts the user when the application is about to load a native library in another situation.

4.4 Automatic Embedding of policies

Enables to compare an application's behavior to a policy stated as a sequence of events rather than a single event, such as a single access to private data, a single call to a system service, or a single execution of a system function. It intend to automatically integrate an arbitrary user-defined policy expressible in an automaton into an application's code.

5 Related work

With the growing popularity of Android and the growing malware threat it is facing, many approaches to securing Android have been proposed recently. Many of the traditional security approaches adopted in desktops have been migrated to mobile phones in general and Android in particular. Probably the most standard approach is to use signature-based malware detection, which is in its infancy when it comes to mobile platforms. This approach is ineffective against zero-day attacks, and there is little reason to believe that it will be more successful in the mobile setting. Program analysis and behavioral analysis have been more successfully applied in the context of Android.

6 Conclusion

This study provides an overview of Android's basic security mechanism, as well as the building elements upon which the application installation policy is based. There are other issues in the installation security policy highlighted here. Furthermore, certain essential and useful policy alterations have been discussed, which can be researched further in the future for the enhancement of the Android system's installation security policy.

References

[1] Sven Bugiel, Stephen Heuser, and Ahmad-Reza Sadeghi. Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 131–146, Washington, D.C., August 2013. USENIX Association.

[2] Michael DeGusta. Android orphans: Visualizing a sad history of support. *Message posted to http://theunderstatement. comlpostl11982112928/android-orphans-visualizing-a-sadhistory-of-support*, 2011.

[3] William Enck. Defending users against smartphone apps: Techniques and future directions. In *International Conference on Information Systems Security*, pages 49–70. Springer, 2011.

[4] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 3–14, 2011.

BFH1925019F

ORIGINALITY REPORT

52%	44%	11%	18%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	docplayer.net Internet Source	19%
2	David Barrera, Jeremy Clark, Daniel McCarney, Paul C. van Oorschot. "Understanding and improving app installation security mechanisms through empirical analysis of android", Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices - SPSM '12, 2012 Publication	8%
3	www.techrepublic.com Internet Source	6%
4	slidelegend.com Internet Source	5%
5	Submitted to University of Adelaide Student Paper	4%
6	www.usenix.org Internet Source	2%

7	Submitted to Sri Lanka Institute of Information Technology Student Paper	2%
8	global.oup.com Internet Source	1%
9	nsrc.cse.psu.edu Internet Source	1%
10	www.infotech.com Internet Source	1%
11	publica.fraunhofer.de Internet Source	1%
12	www.readkong.com Internet Source	1%
13	www.businessinsider.de Internet Source	1%
14	www.coursehero.com Internet Source	1%
15	Supriya S. Shinde, Santosh S. Sambare. "Enhancement on privacy permission management for Android apps", 2015 Global Conference on Communication Technologies (GCCT), 2015 Publication	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

BFH1925019F