COMPUTERS & SECURITY 77 (2018) 472–487

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Dypermin: Dynamic permission mining framework for android platform

Christos Lyvas[a,*], *Costas Lambrinoudakis*[a], *Dimitris Geneiatakis*[b]

[a] *Department of Digital Systems, University of Piraeus, Greece*
[b] *European Commission, Joint Research Centre (JRC), Cyber and Digital Citizens Security Unit, Via Enrico Fermi, Ispra 2749, 21027, Italy*

ARTICLE INFO

Article history:
Received 7 May 2017
Revised 22 April 2018
Accepted 3 May 2018

Keywords:
Android security
Android permissions map
Reflection technique
Android privacy
Dynamic analysis

ABSTRACT

The Android architecture introduces to the application layer a permission based access control model for restricting access to sensitive phone resources. In this model the access to Application Programming Interfaces (APIs) is protected through permissions defined by the Android OS. The developers in order to utilize protected API methods must declare, in the application's manifest, the appropriate permissions. The "relation" between framework method and a permission can be found through Android's documentation. However, not only documentation may accidentally lack information but also Android features undocumented and hidden API methods. Undoubtedly a major challenge for researchers today, *is the accurate identification of API methods and permissions pairs*, which compose the permission map for the Android framework.

This paper introduces Dypermin; a transparent framework for compiling the Android permission map without requiring any modification to the underlying operating system. To achieve that, Dypermin capitalizes on intrinsic properties of the Android framework that is security exceptions during runtime and the availability of any protected API method through the Android framework, as well as on the advantages of Java reflection mechanism. Dypermin, in contrast to other related methods, validates itself as it relies on runtime information, meaning that it does not generate false positive map entries. Dypermin has been evaluated on different Android versions. The results have been compared with the respective results of other proposed methods in order to demonstrate Dypermin's efficacy for compiling the Android permission map for any given version.

1. Introduction

Android undoubtedly is considered the dominant operating system (OS) in the smartphone market IDC: Smartphone OS Market Share. Its open source nature constitutes it not only the most preferable OS for mobile device vendors but recently it has been also adopted for general purpose Internet of Things (IoT) devices like smart TVs, Android wearables, etc.

One pillar of Android's security is the process isolation at the kernel level, so that malevolent applications and services do not to affect the reliability of other services/applications or even of the device itself. Furthermore, it introduces an access control model, at application layer, for restricting access to "sensitive" resources (camera, location data, network, to

* Corresponding author.
E-mail addresses: clyvas@unipi.gr (C. Lyvas), clam@unipi.gr (C. Lambrinoudakis), dimitrios.geneiatakis@jrc.ec.europa.eu (D. Geneiatakis).
https://doi.org/10.1016/j.cose.2018.05.007
0167-4048/

mention a few) that could affect user's privacy or cause a security incident. Specifically, access to any sensitive resource is granted through a protected Application Programming Interface (API) method. An application in order to use a protected API method it must first declare the corresponding permissions in its manifest, and request it also at runtime if it is executed on Android latest versions, otherwise a security exception is raised. In any case, users should give their consent for the permissions requested by the application either during the first time that a protected API method is invoked or during installation process, depending on the Android version. A more detailed analysis of the Android security model and the evolution of its permissions subsystem can be found in Enck et al. (2009); Zhauniarovich and Gadyatskaya (2016).

Programmers get information on the correlation between permissions and protected API methods through Android's documentation. To this direction, an issue that attracts the attention of researchers, developers and Android enthusiasts is the question *"What is the exact correlation between Android Software Development Kit (SDK) API methods and permissions?"*. This is caused by the fact that (a) documentation may accidentally lack information, and (b) Android has hidden and internal API methods that are not directly accessible at the application layer since they are not included in the Development SDK. Though the latter cannot be directly accessed there are several publicly available sources that give guidelines on how to gain access to such resources (Android Hidden API, Li et al. (2016a)). So eventually, programmers can gain access to these hidden API methods.

At this point it should be stressed that an accurate correlation between permissions and API methods is of high importance, as this correlation is utilized for malware detection Arp et al. (2014) and other misconfigurations (i.e. overprivileges Geneiatakis et al. (2015)) identification. Today there are attempts, such as Stoaway Felt et al. (2011), PScout Au et al. (2012), Axplorer Backes et al. (2016) and Bartel et al. (2014), to compile the correlation of API methods–permissions that extend Android's documentation. However, these approaches are bounded to specific Android versions and also require access to the underlying OS source code.

Moreover, as the Android OS evolves and in order to improve end-users' experiences, it proceeds with various modifications to the underlying subsystems. For instance, as already mentioned, permissions are enabled dynamically on the latest versions of Android, while from version 6.0 backwards they were granted statically. In addition, some API methods are deprecated, while other are introduced to support additional functionalities. So it is evident that these types of changes not only affect the API methods and permission mapping but also introduce inconsistencies in it among different Android versions according to Zhauniarovich and Gadyatskaya (2016). Thus even other solutions, such as DPSpec Bogdanas (2017), that exclusively rely on annotations (e.g., of documentation) cannot provide a complete coverage for the Android permission mapping.

In this work, we elaborate on the developments of Android's API methods–permissions mapping by proposing a framework, called Dypermin, capable of generating the permission map in a transparent way without requiring access to the OS source code and without generating false positive alarms. Dypermin automatically invokes Android's publicly accessible and hidden API methods in order to intentionally raise runtime security exceptions and thus decide whether or not a permission dependency exists.

More specifically, Dypermin relies on the simple observation that the Android OS raises a security exception if a protected API method is invoked without the appropriate permissions being defined in the application's manifest. Dypermin, in order to identify and report the permissions for every available API method, builds a single application that is automatically invoked after installation.

Dypermin achieves to extract all available API methods since it is provided through the SDK and thus it does not require any modifications to the underlying OS. Furthermore, it validates its finding as it relies on runtime information. Dypermin is evaluated with some well-known classes for different Android versions and its results are compared with both Android public and SDK source documentation as well as with the results of other related proposed methods. Currently, we do not provide a full mapping since it takes a substantial amount of time (see Section 5). It has been proved that Dypermin can accurately identify the API methods–permission map and deduce whether the available documentation is missing a relation between an API method and a permission.

Summarizing, the contributions of this work are:

- The Dypermin framework is capable to compile the Android API methods–permission map without requiring modification of the underlying OS and without generating any false positive pairs of protected API methods–permissions.
- Provides a comparative analysis among Dypermin, other proposed related methods and the Android documentation.
- Dypermin's proof of concept implementation[1] and the results[2] of our analysis are made publicly available.

To the best of our knowledge this is the first work that identifies, in a completely transparent and highly accurate way, the relationship between a given API method and the related permissions without requiring access to the Android OS source code; indeed it requires access to Android SDK, which however, can be retrieved from any Android OS device. We argue that Dypermin is an orthogonal solution to existing ones functioning complementary in order to achieve the highest possible coverage.

The rest of this paper is structured as follows. Section 2 presents background information for the Android OS, while Section Section 3 provides an overview of the related work. Section 4 presents the Dypermin design while its effectiveness is evaluated in Section 5. Dypermin discussion is further elaborated in Section 6 by introducing a comparative analysis with other related works. Finally, Section 7 provides the conclusions and provides pointers for Dypermin's future improvements.

[1] https://github.com/xphctos/dypermin.
[2] https://github.com/xphctos/dypermin-results.