Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 78 (2016) 26 – 32

Procedia Computer Science

International Conference on Information Security & Privacy (ICISP2015), 11-12 December 2015, Nagpur, INDIA

Android Applications Repackaging Detection Techniques for Smartphone Devices

Sajal Rastogi[a], Kriti Bhushan[a], B. B. Gupta[a]*

[a]*Dept. of Computer Engg., National Institute of Technology Kurukshetra, Kurukshetra, 136119, India*

Abstract

The problem of malwares affecting Smartphones has been widely recognized by the researchers across the world. Majority of these malwares target Android OS. Studies have found that most of the Android malwares hide inside repackaged apps to get inside user devices. Repackaged apps are usually infected versions of popular apps. Adversaries download a popular Android app, and obtain the code using reverse engineering and then add their code (often malicious) to it and repackage and release the app. A number of techniques proposed in research and a number of commercial anti-virus products focus on detecting malwares. This is the traditional approach and requires a signature database. Zero day threats cannot be caught with such methods. There are many techniques which focus entirely on detecting repackaged apps. Since repackaged apps are in the majority among the infected Android apps, they can save the user from a large percentage of Android malwares. Detection and prevention of repackaging is also beneficial for original developer/publisher as they do not incur harm to revenue or reputation.

In this paper, we study in detail about some of the repackaging detection techniques. Mainly, there are two kinds of techniques - offline and online. They serve different purposes. An offline technique cannot be replaced by an online technique and vice versa. Offline techniques are for direct use of app market owner, whereas online techniques are for direct use of Android users. We study different offline and online techniques. These techniques use different features and metrics to detect similarity of apps and they are representatives of their category of techniques.

Peer-review under responsibility of organizing committee of the ICISP2015

Keywords: Android; Smartphone security; repackaging; cloning; app similarity

* Corresponding author. *E-mail address:*gupta.brij@gmail.com

Peer-review under responsibility of organizing committee of the ICISP2015
doi:10.1016/j.procs.2016.02.006

1. Introduction

Android is the most targeted smartphone OS. According to F-Secure, an incredible 97% of new mobile malware families are targeting Android[1]. In only the first quarter of 2014, 275 new Android threat families were identified by F-Secure[2]. The number of new threats identified for other smartphone OSs was ignorable compared to this figure. Studies[3,9]have made a very useful observation that most of the Android malwares, 86% of malwares as per[3], and 73% of malware families as per[9], use repackaged apps as the medium of propagation and installation. Repackaging an app with a malware is easy, and the popularity of original app helps the malware in infecting a lot of devices quickly. It has been found that many apps are repackaged to redirect the advertisement revenue from the original publisher to the adversary[12,17,20].

The existing techniques capable of detecting app repackaging can be classified as offline and online. Offline techniques are those that can be used for vetting app markets. Offline techniques detect repackaged apps among millions of apps from one or more market(s). Scalability becomes a more desirable trait for these techniques than accuracy. Online techniques are those that perform a significant part of their job on the user device. They usually detect whether an app is repackaged at the installation time. There may be some modifications that apps need to go through before installation for the online techniques to be effective. We discuss both kinds of techniques in this paper.

This paper is composed of the following sections. Section II introduces Android security, app repackaging, and the techniques to detect repackaging. In section III, we shed some light on Android OS, app repackaging, and app repackaging detection. Section IV discusses various techniques that claim to detect repackaging and highlights their unique features. Section V then presents the key takeaways from section IV. Finally, section VI conclude this paper and discuss some scope for future work.

2. Android app repackaging

During repackaging of apps, modifications can be made to the app by the adversary (plagiarist). These modifications performed may be one or more of the following: replacing of an API library with adversary owned library; redirecting the ad revenue of the app if the app uses some ads; adding some ads to the app; introducing malware code inside existing method(s); adding method/class specially for introducing malware code.

After the necessary modifications, the adversary can prepare a package (APK file) again. The adversary signs the app with her private key and the public key in the META-INF directory now corresponds to this private key. This app is now released on some unofficial market where the user fall prey to it.

Some repackaging detection/deterrence solutions assume that the adversary wants to exploit the popularity of the original app to infect a large number of users quickly. Thus, they work on the assumption that the metadata of the repackaged app is very similar to that of the original app. On the other hand, some solutions assume that the adversary is repackaging an existing app because she wants to save time/effort of creating a host app for the malware. In this case, the adversary can significantly change the metadata in her repackaged version. The only way to detect similarity in such cases is to compare the functionality/code of each and every pair of apps. The third possible case in which even the functionality is changed cannot be called repackaging.

3. Android app repackaging detection techniques

This section presents some of the better techniques that have been proposed by the researchers for detecting repackaged apps. An important thing to understand is that a technique does not have to be perfect. If a technique forces the adversary to apply many obfuscations/modifications, and makes the cost of repackaging high enough that the adversary makes no profit, then it is more than satisfactory.

3.1. AnDarwin

Crussell et al.[4] present AnDarwin, an offline tool. Scalability is a pre-requisite of any offline tool. Scalability is, indeed, the primary focus of the creators of AnDarwin. AnDarwin boasts of a sub-quadratic time complexity by