Journal of Information Security and Applications 40 (2018) 20–30

Contents lists available at ScienceDirect

Journal of Information Security and Applications

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

Instance based security risk value estimation for Android applications

Mahmood Deypir[a], Abbas Horri[b]

[a] Shahid Sattari Aeronautical University of Science and Technology, Tehran, Iran
[b] Department of Computer Engineering, Shahrekord University, Shahrekord, Iran

ARTICLE INFO

Article history:

Keywords:
Android security
Malwares
Instance based risk metric
Decision making

ABSTRACT

Android has emerged as the widest-used operating system for smartphones and mobile devices. Security of this platform mainly relies on applications (apps) installed by the device owner since permissions and sandboxing have reduced the attack surface. Android antivirus programs detect known malware based on their signature, but they cannot detect zero-day viruses. Therefore, estimating security risk could be helpful for comparing and selecting apps that are more likely to be malicious or benign based on the estimated risk values. Therefore, systematic assistance for making appropriate decisions can significantly improve the security of Android-based devices. Additionally, Android markets can leverage estimated risks to recognize suspicious apps for further analysis. In this study, a new metric is introduced for effective risk estimation of untrusted apps. While previously proposed risk measurements are based on features such as permissions and function calls, our devised metric benefits from previously known malicious and non-malicious app instances. The metric uses previously identified malware and normal app samples to compute the security risk of untrusted apps. Thus, previously known samples are represented in the feature space, and for each untrusted input app, the risk is estimated using distances to malicious and non-malicious app instances. Moreover, to increase the metric's detection rate, an instance and feature weighting schema is suggested. Empirical evaluations on various datasets show that the proposed instance-based metric has higher detection rates and is more effective than a previously proposed feature based on risk score measurements.

1. Introduction

Most capabilities and features of Android based devices for end users rely on applications (apps) that can be installed on these devices. Anonymous developers or app markets can provide apps. Access to various types of services and sensitive information including private personal data, contact lists, geolocation, sent and received messages, and social networks can be provided in Android apps. These types of accessibilities, functionalities, and facilities make privacy and security issues more challenging. The security architecture of this operating system reduces the attack surface by restricting applications using permissions and sandboxing. Therefore, in order to perform malicious activities such as stealing user data, sending premium messages, and making phone calls, an attacker must deceive users into installing a malicious app since other ways of intrusion are nearly closed in the Android system. When installing an app, Android requires the user to grant privileges through requested permissions. Many applications are developed for this operating system, which requires various permissions based on functionalities. Application permissions are displayed in the first screen of the installation program. The end user of an Android based mobile device must approve these permissions or discard them to install the application. The privileges remain unchanged until they are revoked from the app when the user issues the app removal process. Although this security mechanism is very simple and straightforward for users, it creates many challenges. First, a user usually does not spend much time studying permissions and thinking about their effects, so they tend to go forward and to complete the installation process. Moreover, an ordinary user does not have the technical skills about Android permissions and their impacts. Furthermore, an ordinary or malicious app may request similar permissions, making it challenging for end users to make a correct decision. Therefore, this security model is ineffective regarding the end user's security and privacy to preserve their personal information from disclosure or to prevent monetary resource abuse by various types of potential malware. Consequently, an Android malware, e.g., spyware, Trojan, or Adware, can deceive users by introducing itself as a useful app and steal a user's personal or business data as well as use their mobile phone credit and money. There exists some research on enhancing the Android security model and its security risk communication mech-

E-mail addresses: mdeypir@ssau.ac.ir (M. Deypir), ahorri@eng.sku.ac.ir (A. Horri).

https://doi.org/10.1016/j.jisa.2018.02.002
2214-2126/© 2018 Elsevier Ltd. All rights reserved.

anism [1–6]. Examples of such security efforts include using better and intuitive titles for permissions, categorizing permissions based on their effects, reducing the number of permissions by merging similar permissions, utilizing user reviews about apps, and using visual security indicators for risky apps. Additionally, several approaches have been proposed to measure Android apps' security risk [2,10,31,23]. Based on an effective security risk measure, it is possible to compute the security risk of an app and fire a warning signal to the user if the computed risk exceeds a predetermined threshold. Moreover, users can compare similar apps functionalities in term of their risk scores. Furthermore, Android markets require an effective risk computation metric to identify suspicious apps among the vast number of newly submitted apps by developers for further examination. Detailed analysis and deterministic malware detection for each app is very time consuming, and systematic filtering of low risk apps is an important requirement. However, our evaluations show that current measures and models of Android risk computation have unacceptable performance that could make end users and Android market trust them. That is, they don't compute relatively high risk values for known malware and low risk quantities for benign apps well enough to recognize malicious apps. In this paper, we propose a new security risk score measurement that performs better than previously proposed ones. While previous risk measures are mainly feature based, the proposed approach is instance based since it benefits from previously known malware and goodware samples represented in the feature space. It can leverage various feature type including both static and dynamic ones. We have shown the proposed metric's effectiveness through extensive experiments on many actual Android app samples including both malware and goodware. This paper is organized as follows: In the next section, previous research regarding Android security and malware detection are reviewed. The problem statement is presented in Section 3. In Section 4, the formulation of our new security risk score metric is introduced, and related algorithms for computing and using it are described. Extensive experimental evaluation of the proposed measure with respect to previously proposed ones are presented and illustrated in Section 5. These experiments have been performed using known malware in the Android world and ordinary useful apps belonging to the Google App store. Finally, Section 6 discusses related issues and concludes the paper.

2. Related works

Quite a bit of research exists regarding the security of mobile devices based on Android security architecture. Mobile phone users participate in device security by approving requested permissions of an app or declining the permissions, which is equal to canceling the installation process. Research findings show that most users disregard permission checking requested by an Android app. Researchers are trying to overcome this problem and thus enhance the Android security architecture [3–6]. Felt et al. [3] proposed solutions like changing the categorizations of Android permissions, emphasizing the security risk instead of permissions, and approving permissions via a specific method. Kelley, Cranor, and Sadeh suggest that a high level critical information access regarding user privacy including personal data, location information, and contact lists should be displayed instead of the permission names in the first installation page [7]. Gates et al. recommend visualizing summary risk and safety scores in order to reduce the required space for displaying permissions and assist the user for fast and effective decision making [1]. Such scores are quantities computed based on various permissions requested by an app. For most users, displaying a summary of risk or safety scores by graphical indicators are more effective than textual information of the permissions in term of user notification. Peng et al. [8] introduce statistical measures and mining models to compute security risk scores and rank apps based on the requested permissions. Their approach ranks applications in an Android app store like Google Play based on security risk values. Such a ranking encourages users to select more secure apps when there are several apps with the same functionality and different risk scores. Moreover, similar definitions were introduced for the concept of security risk regarding the list of permissions requested by apps. Gates et al. [2] extend Peng et al.'s [8] work by precisely describing many statistical and probabilistic generative risk scores for Android apps using permission usage patterns. All measures are defined based on critical permissions, which is defined as a permission that can access sensitive software and hardware mobile resources and their usage patterns in malicious apps. Android malware usually abuses critical permissions and corresponding *API* functions within its code to perform a malicious activity. Gates et al.'s [2] proposed risk scores are generative and mainly computed using benign app permission usage information. However, to improve performance, the authors increase the impact of some critical permissions on the resulting risk score values. They manually selected nine critical permissions that can be misused by malware. Sarma et al. [10] uses critical permissions rarely requested by normal apps to recognize risky apps. Grace et al. [31] introduced an automated system called *RiskRanker* to examine whether an app possesses known dangerous behaviors. For this purpose, it uses common exploits used in malwares or family of them belonging to a specific period of time. Since deterministic detection of zero-day malware requires further analysis, the system can be used as a preprocessing step to sift through a plethora of apps from an Android market by producing a prioritized list of suspicious apps based on their computed security risk. Enck et al. [23] developed the Kirin system to examine combinations of risky permissions to determine whether permissions requested by an app satisfy a certain global safety policy. Permission combinations e.g., WRITE_SMS and SEND_SMS, are manually specified in this system. These combinations could be used in a malicious apps and therefore are used to identify malware.

Literature has proposed several approaches to classify Android apps into malware and benign apps [9–11]. The aim is to construct a mining model like naïve bayes, based on labeled apps augmented by information regarding the static and dynamic behavior of malware and clean apps to classify future malware. Other research uses static code analysis of decompiled apps to analyze malware's malicious activities and behaviors. In this approach, permission to function map is performed as a preprocessing step that recognizes which function calls are used and what their ordering is. For example, some malware conducts malicious behaviors such as accessing the contact list or storage and sending a SMS. In static code analysis, the extracted knowledge and patterns are used to distinguish malicious apps from ordinary applications [12–15]. Malware detection and risk score computation based on static source code analysis can be regarded as a complementary method for permission analysis. However, it faces challenges like code obfuscation and code writing techniques exploited by malware writers. Desnos proposed a distance based algorithm for similarity measurement among Android apps [12]. This algorithm can be utilized for malware detection and risk computation. Schmidt et al. proposed a collaborative architecture for Android malware detection using static code analysis [13]. In this architecture, Android based devices collaborate together and communicate to a remote server to analyze apps and to detect malwares. Similar to *RiskRanker* system described above, approach devised in [14] uses previous malware footprints as well as some heuristic for detecting new unknown malicious apps. For malware detection, Aafer et al. [15] proposed an approach in which a couple of well-known classifiers were used and tested on extracted static feature of available malwares and benign apps. Dynamic behavior analysis of running An-