Journal of Information Security and Applications 41 (2018) 144–158

Contents lists available at ScienceDirect

Journal of Information Security and Applications

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

iABC: Towards a hybrid framework for analyzing and classifying behaviour of iOS applications using static and dynamic analysis

Arpita Jadhav Bhatt*, Chetna Gupta, Sangeeta Mittal

Jaypee Institute of Information Technology, Noida 201301, India

ARTICLE INFO

Article history:

Keywords:
iOS applications
Reverse engineering
Machine learning
Static analysis
Dynamic analysis
Static risk score

ABSTRACT

Is this app safe to use? - A wrong decision can result in privacy breach in iOS devices. In this digital era users extensively use smart devices to store their personal and important information. To ease users' tasks, thousands of free or paid apps are available in app store. However, recent studies reveal startling facts about various attacks and data harvesting incidents through these apps, where personal data is put at risk. Through this paper, we propose a permission induced risk model- iOS Application analyzer and Behavior Classifier (iABC), for iOS devices to detect privacy violations arising due to granting permissions during installation of applications. It is a two-layer process comprising of static and dynamic analysis. It uses reverse engineering to extract permission variables from applications and computes a risk score for each application using ranking algorithms. The approach considers application's category as a key feature for detecting malicious applications while computing static risk score. Different machine learning classifiers were employed to evaluate 1,150 applications. The empirical results show that our proposed model gives detection rate of 97.04%. Furthermore, to assess privacy breaches by applications at run time, dynamic analysis on 50 applications has been performed to obtain dynamic risk scores of installed apps.

1. Introduction

Decreasing costs and increasing ease of use has led to surge in number of smartphones, simultaneously leading to the challenges in protecting privacy concerns of users operating them. One of the most commonly used mobile operating system (OS) after Android, is iOS, which is designed and marketed by Apple Inc. [1,2]. iOS SDK provides a platform for application developers to develop, upload and distribute their applications on App store. According to a study done in January 2017 [2], App store had more than 2.2 million iOS applications and their download exceeded more than 140 billion in September 2016.For protecting user's privacy and ensuring that applications are developed by trusted enterprises/developers [3,4], Apple follows a mandatory code signing, encryption and sandboxing procedure to provide security in terms of a guarantee that an application is not altered since it was last signed [4]. However, to make apps usable, iOS allows third-party applications to access user information such as contacts, location, photos, recent searches, device information, address book, geo-coordinates, email-ids, passwords, bank credentials etc. This sometimes is misused by malicious applications which in turn can breach users' privacy.

According to App Store guidelines, apps must ask for user's permission to access sensitive data [5]. Before using phone resources each app has to ask for permissions for the resources which an application tries to access. Though, it never informs the users how, when, with whom and what kind of information will be shared. Moreover, there is no permission control to implement the principle of least privilege for the apps.

Being closed-source, iOS has been considered to be a safer system than its peer Android OS. However, recent malware attacks (such as Dutch ransom attack) [6], worm attacks (such as *ikee*), Wirelurker [7,8], Masque attack [8,9], have proven that iOS devices are also vulnerable. Furthermore, use of code obfuscation and polymorphic coding techniques adopted by developers make the attacks undetectable by pattern matching anti-attack software. A well-known example is of *Jekyll* applications that could bypass Apple's code review process and once installed, it was capable of altering its behavior at run-time [10]. Adware present in application is infamous for collecting user's personally identifiable information for marketing purposes. In view of the recent attacks on iOS devices and presence of malicious applications, many research studies have focused on classifying benign and malicious applications, detecting malware and analyzing permission induced risk associated with a mobile application [11,12]. Apart from this, some researchers have explored the concept of automated analysis systems to analyze iOS applications binaries [13].

* Corresponding author.
E-mail address: arpitajadhav@gmail.com (A.J. Bhatt).

https://doi.org/10.1016/j.jisa.2018.07.005
2214-2126/

However, none of the known approaches identifies privacy breach by an iOS app that may seem benign and is not captured by existing malware detection methods. This paper is concerned mainly with privacy breach by iOS apps without end users' knowledge. It was found by reverse engineering few apps that apps have permissions to access device resources share personal details of users without using encryption. Thus, it is the onus of security feature to determine all possible permissions that the app may seek during its lifetime. Even for permissions duly sought from user, iOS devices may face two types of threats namely, abuse of security-critical private APIs and sharing (uploading without notifying the user) private data to other devices. A set of methods for identifying malicious apps has-been based on static and dynamic analysis of iOS applications. Static analysis is performed either on source code or application binary for data leakage without actually running the source code of the application. Third-party applications for iOS devices are written in Objective-C [14] or Swift [15]. Objective-C's run time functions allow execution-time modification and manipulation, so it is easy to modify applications behavior and inject malware. Therefore, in dynamic analysis, behavior of an application is analyzed for any data leakage during its run-time i.e. on executing the application code. Only static analysis lacks in detection of run-time malicious behavior of applications. Similarly, just dynamic approaches also suffer from incomplete code coverage issues as stated in [13], and thus fails to detect malicious application or private data uploading [16].

To mitigate these privacy-related problems, a breach detection framework for analyzing and classifying behavior of iOS application by using both static and dynamic analysis which has been proposed and implemented by applying reverse engineering and penetration testing. To capture permission induced risk, application's category has been taken into consideration when classifying the application as benign or malicious [11]. Category reflects application's core features. Detecting malicious applications by considering application's category gave better detection rate since malicious intents within a category could be identified more accurately. Apple provides 24 categories for a developer to classify his app based on the utility and features being provided [17].

Use of category information to detect malicious application has not yet been made in any existing work. This will help in monitoring misuse of risky permissions and reduce false positive rates during classification. The presented framework systematically estimates the risk relevance of permissions by employing different ranking algorithms such as correlation coefficient, mutual information and *t*-test as measures of mapping permissions to an application category. The accuracy of detection of malicious applications has been computed using various machine learning classifiers and results indicate that proposed model improves the detection rate of malicious applications by 15.92%, 15.39%, 16.43% and 19.74% for Naïve Bayes, Random Forest, KNN and Linear SVM respectively.

1.1. Main contributions of the paper are listed below

- Hybrid framework for analyzing and classifying iOS application behavior by utilizing application category information to classify them as benign or malicious using static and dynamic analysis.
- One of the first study on quantitative analysis of privacy breach in iOS apps
- An approach to track the behavior of iOS applications by combining reverse engineering and penetration testing to uncover privacy risks in the app.
- A systematic risk ranking assessment approach with respect to permissions sought by an application at any point in time.
- Devise criteria for dynamic analysis capable of capturing dynamic malicious behaviour of an iOS app with respect to privacy breach.

The rest of the paper is organized as follows: Section 2 discusses background work followed by Section 3 that describes proposed malicious intent detection model for an iOS application. Section 4 elaborates our approach to static analysis of applications. The dynamic malicious behavior of apps has been defined and analyzed in Section 5. Paper has been concluded in Section 6.

2. Background work

Previous work on static and dynamic analysis has mainly focused on desktop operating systems and was not readily implementable on mobile operating systems like Android, Blackberry, windows, iOS [13]. Android operating system (OS) is one of the most widely used mobile operating system as it is an open platform. In the past few years Android has become a major target of malicious applications [11] and therefore most of the studies have focused on detecting malicious applications for Android OS. In case of iOS devices, Apple has designed them in such a way that it does not need antivirus to secure its devices [18]. However past attacks due to third-party applications which were capable to bypass Apple's code review process as well as attack non-jailbroken devices have demonstrated that Apple's security measures are insufficient to secure its devices. In this paper, we focused our study towards analyzing benign or friendly applications for their extent of privacy breaches. Due to lack of availability of the source code of iOS applications, and since they are distributed as binaries on App store [16], the applications were reverse engineered to extract necessary frameworks, classes, and methods.

Reverse engineering is a powerful technique to provide crucial information about the behavior of an application and discover vulnerable code that can be compromised at run-time by the malware. Reverse engineering helps an analyst to scrutinize the application when source code is unavailable [19]. Testers or security analysts follow reverse engineering approach to identify bugs in the application, check its functionalities and analyze the general flow of the application [19].

2.1. Static analysis of mobile apps

Egele et al. have developed a tool PiOS [20] that automatically analyses iOS applications behavior. Their study took into account the threats iOS applications may pose to user data. Comprehensive control flow graphs have been plotted from application's binaries. After that, reach ability analysis was done on the generated CFGs and private data leaks were identified. Tim Werthmann et al, have developed a working prototype of the framework which is embedded in a tool named as PSiOS [21]. The tool is based on mobile control flow integrity for iOS devices. PSiOS also enables user-driven as well as fine-grained sandboxing for iOS applications. Their research work aims to address the problem of discovering privacy leaks for iOS applications and preventing them from assigning explicit sandboxing profiles for every application. Their proposed approach warns users when a malicious app tries to access private data.

Enck et al have [22] proposed a framework called as *Kirin,* which incorporates install time certification mechanism. It allows the smartphone device to enforce a list of predefined security requirements before installing the application i.e. during the installation of any third-party application, Kirin framework informs the users regarding the resources, which the application would be accessing. The framework is utilized when a user initiates installation process for an application package. It uses action string with