International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.3., Issue.2, 2015

REVIEW ARTICLE

ISSN: 2321-7758

COMPUTING MALWARE SCORES FOR MOBILE APPLICATIONS (MOBILE COMPUTING)

V.HEMALATHA[1],K.S.HEMAPRIYA[2],P.R.JOSHNA[3], S.T.SANTHANALAKSHMI[4]
[1, 2, 3]Student, Dept. of. CSE,Panimalar Engineering College, India.
[4]Associate Professor, Dept. of. CSE, Panimalar Engineering College, India

Article Received: 18/03/2015 | Article Revised on:31/03/2015 | Article Accepted on:03/04/2015

MECHANICAL
ENGINEERING
www.ijoer.in
ENGINEERS
MAKE A WORLD OF DIFFERENCE
International Journal of Engineering Research-Online

ABSTRACT
Based on the studies of semi automated composite we found that, Web services select concrete services based on the functional and/or non- functional attributes. They do not consider relationships between these attributes in the description of services or the user constraints. Thus, we propose an approach, which relates services to objects (resources) maintained by these services. The user can impose his constraints on the objects affected by the requested services. The affected object and their relationships are described in an intermediate ontology using OWL-DL and SWRL languages. Our selection strategy considers the relationships between services by looking for the dependent instances (conforming objects values) of affected objects that satisfy the user constraints and by combining the related services to get conforming composite services. The proposed selection approach of conforming composite services is implemented and integrated using Ant Colony optimization follow paths with conforming values of concrete services using SPARQL query.

Keywords: Concrete service, OWL, SPARQL query, Ant Colony.

1. INTRODUCTION:

MOBILE devices are becoming ubiquitous, and they provide access to personal and sensitive information such as phone numbers, contact lists, geo location, and SMS messages, making their security an especially important challenge. Compared with desktop and laptop computers, mobile devices have a different paradigm for installing new applications. For traditional personal computers, a typical user installs relatively few applications, most of which are from reputable vendors, with niche applications increasingly being replaced by web-based or cloud services. For mobile devices, one often downloads and uses many applications (or apps) with limited functionality from multiple unknown vendors. Therefore, the defence against malicious applications must depend to a large degree on decisions made by the users. An important part of malware defence on mobile devices is to communicate the risk of installing an app to users, and to enable the user to make informed decisions about whether to choose and install specific apps. We study how to effectively evaluate the risk of mobile applications, with a focus on the Android platform. The Android platform has emerged as one of the fastest growing operating systems. In May 2013 Google Inc. announced that900 million Android devices have been activated. Additionally Google Play (formerly known as Android Market) crossed more than 48 billion downloads, and is now averaging about 2.5 billion downloads per month. Such a wide user base, coupled with ease of developing and distributing applications, makes Android an attractive target for malicious developers that seek personal gain while costing users money and invading users' privacy. One of Android's main defence mechanisms against malicious apps is a risk communication mechanism which warns the user about the permissions an app requires before the app is installed by the user, trusting that the user will make the right decision. The specific approach used in Android has been shown to be ineffective at informing users about potential risks. The majority of Android apps request multiple permissions. When a user sees what appears to be the same warning message for almost every app, warnings quickly lose any effectiveness as the users are conditioned to ignore such warnings. We believe that the main reason for the failure of the current Android warning approach is that it presents the risk information of each app in a "stand-alone" fashion and in a way that requires too much technical knowledge and time to distil useful information. Recently, binary risk signals based on the set of permissions an app requests have been proposed as a mechanism to improve the existing warning mechanism, requesting certain permissions or certain combinations of two or three permissions triggers a warning that the app is risky.

2. OVERVIEW OF EXISTING SYSTEM:

In existing system, risk communication mechanism which warns the user about the permissions an app requires before the app is installed by the user, trusting that the user will make the right decision. The specific approach used in Android has been shown to be ineffective at informing users about potential risks. The majority of Android apps request multiple permissions. When a user sees what appears to be the same warning message for almost every app, warnings quickly lose any effectiveness as the users are conditioned to ignore such warnings.

2.1Drawbacks of the Existing System:

The existing system allows malicious application, reports the risk in standalone manner and warnings quickly lose any effectiveness as the users are conditioned to ignore such warnings.

3. PROPOSEDAPPROACH:

In order to overcome these drawbacks, the concept of risk scoring function which assigns each app a numerical score, that indicates how risky the app is. This approach presents "comparative" risk information that each app's risk is presented in a way so that it can be easily compared to other apps. Given a risk scoring function, one can construct a risk signal by choosing threshold above which the signal is raised. However, we believe that it is better to use a risk scoring function for risk communication in the following way. Given this function, one can compute a risk ranking for each app, identifying the percentile of the app in terms of its risk score. This percentile number has a well-defined and easy-to-understand meaning. Users can appreciate the difference between an apps ranked in the top 1 percent group versus one in the bottom 50 percent. This ranking can be presented in a more user-friendly fashion, e.g., translated into categorical values such as high risk, medium risk, low risk, and very low risk. An important feature of the mobile app ecosystem is that users often have choices and alternatives when choosing a mobile app. If the user knows that one app is significantly more risky than another with similar functionality, then that may cause the user to choose the less risky one. Such an approach complements well other approaches that try to identify malicious apps. After malicious apps are removed, the remaining ones can be ranked according to their risks.

3.1MERITS:

Framework that includes both the rarity based risk signals and probabilistic models, and explore other ways to instantiate the framework. Idea of risk score functions to improve risk communication for Android apps.

4. ALGORITHM AND DESIGN:

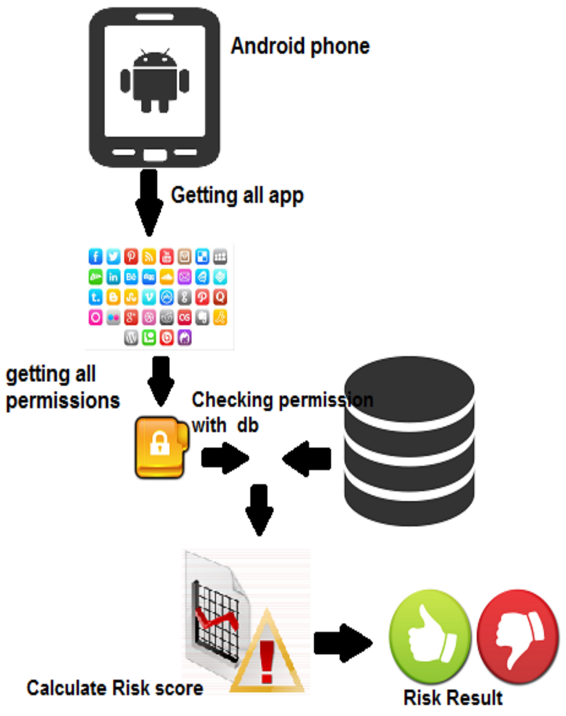
4.1. ALGORITHM:

The method we propose is Rarity Based Risk Score with Scaling, which allows the use of scaling factor to penalize requesting high risk permissions more than requesting lower risk permissions. While the PNB method could have more dramatic impact using the prior, the beta values used to define the prior would have grown exponentially to have significant impact on the outcomes.

$$score(\mathbf{x}_i) = \sum_{m=1}^{M} x_m \cdot w_m \cdot \ln\left(\frac{N}{c_m}\right).$$

Using this equation we are able to scale the importance of permission relative to its risk. So medium risk permission can have wm ¼ 2 times the impact on the overall score, and high risk permissions can have wm ¼ 3 times the impact on the overall score. For the evaluation we use wm values that reflect the values from PNB for each permission. Further fine-tuning the wm values is possible. However, we consider it undesirable to tune wm based on the malware data set because of potential for over fitting. In any case, this would make any comparison with the probabilistic generative models unfair, since they were constructed using only the market data sets.

4.2 ARCHITECTURAL DIAGRAM:



5. IMPLEMENTATION DETAILS:

Getting Installed Apps:

Android has a growing section of third party applications, which can be acquired by users either through an app store or by downloading and installing the application's APK file from a third-party site. The app filters the list of available applications to those that are compatible with the user's device, and developers may restrict their applications to particular carriers or countries for business reasons. But most of the users download the APK files from third party servers and installed into mobiles, Most of the apps from trusted sources are not malware, but the third party server providing malwares in modified APK. So user has the power to list all the apps installed in their mobile, then user can identifies the Application is Risk or not.

Getting Requested Permissions:

Different apps have different functionalities, and thus may require different permissions; it thus makes sense to take into account the intended functionality of an app when deriving a risk signal based on permissions. We use the category of an app to approximate the intended functionality of an app. This is partially supported by the analysis category an app was in affects the permissions it requests. Here we list out all the requested permissions in selected Application.

Calculate Risk Score:

In this module if user select's any running application its Manifest permissions are shown to the user. It can be easy for the user to identify the malware. When risk scores are used to rank apps, using the probabilities is equivalent to using the negative logarithm of the probabilities. The first method we propose is the Rarity Based Risk Score which is based strictly on the fraction of applications which are requesting a specific permission, where the rarity of permissions is the primary indicator that contributes to raising a warning for an app. However, instead of considering only the rarest permission, we accumulate risk across all permissions that the app requests. In this formulation, the higher the score, the more risky the application is. This formulation also only considers permissions that are set when calculating the risk score, unlike the Bayesian methods which will affect the score for both set and unset permissions.

Risk Score Result:

In this module the risk score result is generated as report and Chart. The report displays the risk score, average risk score, and status. The report also displays as a chart using a chart Engine. The Risk report is categorized as Highly Risk, Risk, Normal, Protected, and Well Protected.

6. CONCULSION

We discuss the importance of effectively communicating the risk of an application to users, and propose several methods to rate this risk. We test these methods on large real-world data sets to understand each method's ability to assign risk to applications. One effective method is the RSS method which has several advantages. It is monotonic, and can provide feedback as to why risk is high for a specific app and how a developer could reduce that risk. It performs well in identifying most current malware apps as high risk. This method allows for highly critical permissions and less-critical permissions to affect the overall score in an easy to understand way, making it more intuitive as well as difficult to evade when compared with other models.

REFERENCES

[1]. Google Bouncer. http://goo.gl/QnC6G 2014.

[2]. N. Amor, S. Benferhat, and Z. Elouedi, “Naive Bayes versus Decision Trees in Intrusion Detection Systems,” Proc. ACM Symp.Applied Computing, pp. 420-424, 2004.

[3]. K.W.Y. Au, Y.F. Zhou, Z. Huang, and D. Lie, “PScout: Analyzing the Android Permission Specification,” Proc. ACM Conf. Computer and Comm. Security (CCS ’12), pp. 217-228, 2012.

[4]. D. Barrera, H.G. Kayacik, P.C. van Oorschot, and A. Somayaji, “A Methodology for Empirical Analysis of Permission-Based Security Models and Its Application to Android,” Proc. 17th ACM Conf.Computer and Comm. Security, pp. 73-84, 2010.