GENERATING SUMMARY RISK SCORES FOR MOBILE APPLICATIONS

ABSTRACT:

One of Android’s main defense mechanisms against malicious apps is a risk communication mechanism which, before a user installs an app, warns the user about the permissions the app requires, trusting that the user will make the right decision. This approach has been shown to be ineffective as 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 distill useful information. We discuss the desired properties of risk signals and relative risk scores for Android apps in order to generate another metric that users can utilize when choosing apps. We present a wide range of techniques to generate both risk signals and risk scores that are based on heuristics as well as principled machine learning techniques. Experimental results conducted using real-world data sets show that these methods can effectively identify malware as very risky, are simple to understand, and easy to use.
EXISTING SYSTEM:

One of Android’s main defense 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.

DISADVANTAGES OF EXISTING SYSTEM:

- It allows malicious application.
- It reports the risk in stand alone manner.
- Warnings quickly lose any effectiveness as the users are conditioned to ignore such warnings.

PROBLEM STATEMENT:

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.

SCOPE:

The idea of risk score functions to improve risk communication for Android apps, and identify three desiderata for an effective risk scoring function.
PROPOSED SYSTEM:

We thus propose the concept of risk scoring functions. Such a function assigns to each app a numerical score, which indicates how risky the app is. This approach presents “comparative” risk information, i.e., 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 app 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.

ADVANTAGES OF PROPOSED SYSTEM:

- 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.
SYSTEM CONFIGURATION:

HARDWARE REQUIREMENTS:

- Processor: Pentium –IV
- Speed: 1.1 Ghz
- RAM: 512 MB(min)
- Hard Disk: 40 GB
- Key Board: Standard Windows Keyboard
- Mouse: Two or Three Button Mouse
- Monitor: LCD/LED

SOFTWARE REQUIREMENTS:

- Operating system: Android
- Coding Language: JAVA
- Data Base: SQLite
- Tool: Eclipse
REFERENCE:

Christopher S. Gates, Ninghui Li, Hao Peng, Bhaskar Sarma, Yuan Qi,