International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Email:editorijoer@gmail.com http://www.ijoer.in

Vol.4., Issue.3., 2016
(May-June)

RESEARCH ARTICLE

ISSN: 2321-7758

RISK SCORE ANALYSIS FOR ANDROID APPLICATIONS

VANITA CHAUDHARY[1], DEEPALI SARWADE[1], ASHA KAMBLE[1], SAKSHI BHOSALE[1], MADHAVI DHARMADHIKARI[2], VAISHALI KOLHE[2]

[1]BE Student Of Computer Engineering Department, DYPCOE, Akurdi, SPPU, India
[2]Department Of Computer Engineering, DYPCOE, Akurdi, SPPU, India

ABSTRACT

Malicious applications hide in the sheer number of other normal apps, which makes their detection challenging. Developers are provided with improved tools to detect and react to security vulnerabilities. Although strong security measures are in place for most mobile operating systems, the area where these systems often fail is the reliance on the user to make decisions that impact the security of a mobile device. As Android relies on users to understand the permissions that an app is requesting and the installation decision based on the list of permissions. Proposed system work is for security measure of android applications. Instead of finding whether an application is malware or not after installation, proposed system detects the risk at installation level. SVM classifier is used for the app permissions' classification. System assigns a risk score to each application being installed and displays the summary to user in terms of percentage.

Keywords: Risk communication, Risk score, malware.

1. INTRODUCTION

As mobile devices are becoming increasingly popular for personal and business use it is more important to provide users the ability to understand and control the benefit and risk of running apps on these devices. The Android platform has emerged as the fastest growing smart phone operating system being used by about 200 million devices, with around 700,000 devices being activated around the world daily. The access to privacy and security-relevant parts of Android's rich API is controlled by an install-time application permission system. Every application must declare upfront what permissions it requires. The ubiquitous usage of these mobile devices poses new privacy and security threats. Possible access to personal information by unauthorized parties puts users at risk, and this is not where the risk ends. These devices include many sensors and nearly always with us and it provide deep insights in our digital and physical lives. A person often downloads and uses many apps from various unknown vendors, with each app providing some functionality. This different paradigm requires a different approach to deal with the risk of mobile devices, and offers distinct opportunities.

An important part of malware defense on mobile devices is to communicate the risk of installing an application to users, and to enable the user to make informed decisions about either to choose and install the specific apps or not. The majority of Android apps request multiple permissions. When a user sees what appears to be the same warning messages for almost every app, warnings quickly lose any effectiveness as the users are conditioned to ignore such warnings.

2. RELATED WORK

The phenomenal growth of the Android platform in the past few years has made it a lucrative target of malicious applications (app) developers. There are numerous instances of malware apps that send premium rate SMS messages, track users private data, or apps that, even if not characterized as malware [3][5]. Several rules that represent risky permissions are used to flag apps [1]. However, such an approach is not very effective because it does not take into account the intended functionality of app [1] [9].

An effective risk signal is a signal that has a simple semantic meaning that is easy to understand by both users and the developers, is triggered by a small percentage of apps and is triggered by many malicious apps [9]. Existing approaches consider only the risks of the permission requested by an app and ignore both the benefits and what permissions are requested by other apps, thus having a limited effect. Over privileged applications expose users to unnecessary permission warnings and increase the impact of bug or vulnerability.

Stowaway is one of the tools, which detects over privilege in complied android applications. Stowaway is composed of two parts - a static analysis tool that determines what API calls an application makes, and a permission map that identifies what permissions are needed for each API call [11].

At the core of Android security, security model is permission -based system that by default denies access to features or functionality that could negatively impact the user experience, the system or other applications installed on devices. Examples of these features are sending messages or making phone calls, which may incure monetary cost to the user [8].

In Android an application, must request a specific permission to be allowed access to a given resources. Android warns the user about permissions that an application requires before it is installed, with the expectation that the user will make an informed decision [10]. The effectiveness of such a defence to a large degree on choices is made by the users. Therefore, an important aspect of security on mobile devices is to communicate the risk of installing an app to users, and to help them make a good decision about whether to install a given app.

Access control lists (ACLs) and permission – based security models allow administrators and operating systems to restrict actions on specific resources. The main problem with ACLs and permission models in general is they are typically not designed by the users who will ultimately use the system.

The Problem with these permission-based systems is that they are not designed with usability in mind [8].Today's smart phone operating systems, frequently fail to provide users with adequate control over and visibility into how third-party applications use their private data. These are said to be as shortcomings with TaintDroid, an efficient, system-wide dynamic taint tracking and analysis system capable of simultaneously tracking multiple sources of sensitive data. Taint Droid provides real time analysis by leveraging Android's virtualized execution environment. Taint Droid incurs only 14% performance overhead on a CPU-bound micro-benchmark and imposes negligible overhead on interactive third-party applications [6].

Applications that request more permission than needed fit into two general sets: applications that "do something 'questionable' or 'malicious' on purpose" (e.g.,quietly collect information for advertising purposes), or applications that in advertently request additional permissions due to lack of understanding, laziness, or expected future use of a capability. In the first case, the mobile application actually requires permissions that may seem unnecessary to a user [4].

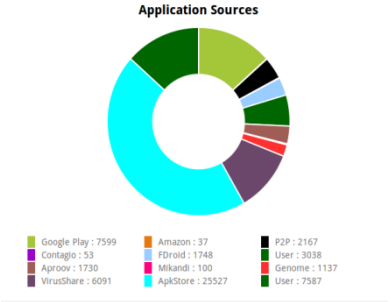


Fig 1. Sources of App Download [11]

3. SYSTEM MODEL

3.1 SYSTEM ARCHITECTURE

Architecture consists of different blocks such as input apk file, check requested permission, SQLite DB, Check threshold, calculating score.

.apk file of the application is taken as an input by first block which user is trying to install. This .apk file is then extracted and provided for further checking.

This extracted list of permissions is taken as an input by next block and it chooses the set of permissions to be processed for risk score calculations. This set of permissions is then considered for all the further calculations required. Permissions are shortlisted and provided to the next block.

Calculating Score: This is an integral part of the system. It interacts with more than one system unit and provides the essential scoring. Using Risk Score Function and SQLite DB the score is calculated. Calculated score is checked for threshold value, and accordingly the application is categorized.

SQLite DataBase (DB): This involves android based database. It stores the permissions which are widely used to invoke malicious applications. It also contains the statistics of these permissions. This data caused for the primary calculations on the permissions.

Various permissions and their packages are extracted for processing. Digital signature for getting access needs to be extracted. From Google play store, user will be downloading the apk file. The system will make use of apk parser and it will scan the system and extract required data. It can add various package names. Digital signature need to be checked. Various rule lists or combination of different rules can be added.

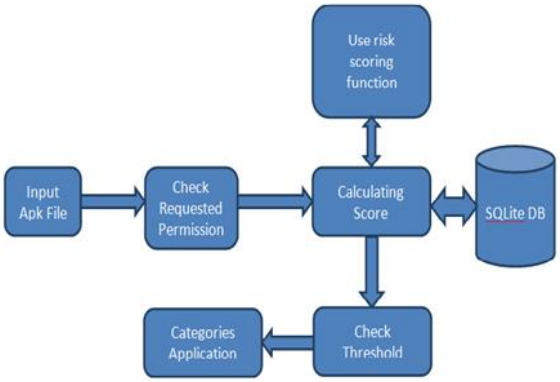
In this system SVM classifier is used for classifying the permissions required by different applications into different categories such as very high risk, high risk, and low risk permissions. This classification will be done on the previously available data of permissions.

Risk score is assigned to the each permission, used by the application. Overall risk score of the application can be found out by calculating average of risk score of all the permissions required by the particular application

Risk score can be calculated on the basis of predefined risk criteria for that single permission. for e.g. if my application using Bluetooth and Bluetooth having risk 0.5% then it will consider in total percentage of risk score.

Various alerts are shown to the user. Percentage of an application being malicious is notified to user. This notification regarding the risk of the particular application is calculated according to overall risk score of the permissions required by that application.

This alert is shown after downloading any application but before installing it so it will be helpful for the user to decide whether to install that application or not**.** Thus user will know about the most probable malicious application and help him to keep his device secure.



SVM (LINEAR CLASSIFIER)

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyse data and recognize patterns, used for classification and regression analysis. SVMs are helpful in text and hypertext categorization.

SVM gives high accuracy, nice theoretical guarantees regarding over-fitting, and with an appropriate kernel they can work well even if you're data isn't linearly separable in the base feature space.

Especially popular in text classification problems where very high-dimensional spaces are the norm.

Maximal Margin Hyperplanes

If the training data are linearly separable then there exists a pair (w, b) such that

$w^T x_i + b \geq 1$, for all $x_i \in P$

$w^T x_i + b \leq -1$, for all $x_i \in N$

with the decision rule given by

$f_{w,b}(x) = \text{sgn}(w^T x + b)$.

w is termed the weight vector and b the bias (or − b is termed the threshold). The inequality constraints can be combined to give

$y_i (w^T x_i + b) \geq 1$, for all $x_i \in P \cup N$

Without loss of generality the pair (w, b) can be rescaled such that

$$\min_{i=1,\ldots,l} |w^T x_i + b| = 1 ,$$

this constraint defines the set of canonical hyperplanes on R^N.

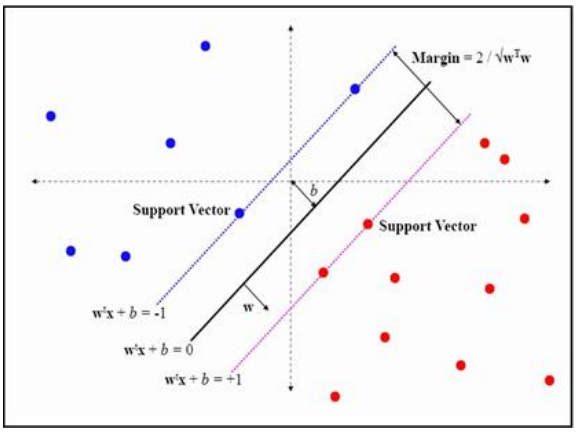


Fig.3.SVM Hyperplane

3.1 RISK SCORE AND RISK SIGNAL

Two relevant measures of risk signal are the warning rate which defines how often a user receives warnings generated by the risk signal and the detection rate which defines what percentage of malicious apps will trigger the signal.

To avoid over-exposing users to warnings generated by risk signals, it is desirable that a risk signal has a low warning rate. To be effective at detecting malicious applications a risk signal should have a high detection rate. Moreover a risk signal should be easily understandable by end users. After all, no risk signal can be used to stop the installation of an app by itself. The ultimate decision lies with the end user. If the user can understand why a warning is raised, then there is higher chance that they can process the information accordingly. Having an easy-to-understand risk signal also has the potential to benefit the overall eco-system of Android apps.

A risk score for apps based on their requested permission sets and categories

$$\text{rrank}(ai) = \frac{|\{a \in A | \text{rscore}(ai) \geq \text{rscore}(ai)\}|}{|A|}$$

Where a_i represents the dataset.

The above gives a risk ranking relative to all apps in all categories. An alternative is to rank apps in each category separately, so that one has a risk ranking for an app relative to other apps in the same category.

Rare critical permissions (#RCP (Θ)): a critical permission is rare with respect to a threshold Θ if it occurs in less than Θ percent of the Android Market applications.

Rare pairs of critical permissions (#RPCP (Θ)): a pair of critical permissions to be rare with respect to a threshold Θ if the individual permission's frequency is above threshold Θ but the frequency of occurrence of the two permissions as a pair is below Θ, and this can be defined as (#RPCP (Θ)).

3.3 DATASETS

Market data sets- two data sets have been collected from Google Play spaced three years apart. The data sets are collected by crawling the Google Play app store, starting with the top apps in each category and then following all the links to other apps on each page.

Market2011, the first data set, consists of 157,856 apps available on Google Play in February 2011. Market2014, the second data set, consists of 324,658 apps, and has been collected in February 2014. It can be assumed that apps in these two data sets are mostly benign. While it can be believed that a small number of malicious apps may be present in them, assuming that these data sets are dominated by benign ones. Market2014 data set is used for model generation and testing, and Market2011 data set is used for validation and market evolution analysis.

Malicious apps request permissions in different ways from normal apps, and it indicates

that looking at permission information is in fact useful. It also shows that there may be a slow evolution in the market data set.

4. RESULT

Application permissions are checked according to the permission set provided by the system developed. Permission set of the existing applications is parsed. Permissions are classified using SVM classifier. Risk score is calculated according to permissions used by the application. Alert message is shown to user.

Fig 4(a) shows the basic UI. Rules are added to check the permissions used by the application and accordingly risk score can be generated.

Permissions will be checked before installing any application and risk score will be shown to the user. User will decide whether to install an application or not.

Fig 4(b) shows installation of the application. After that risk score is calculated and prompted to the user as shown in Fig 4(c)

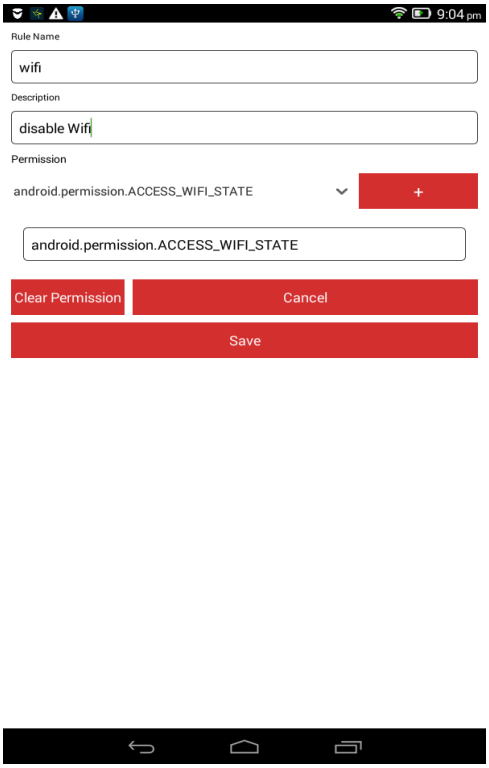


Fig. 4(a). Adding rules

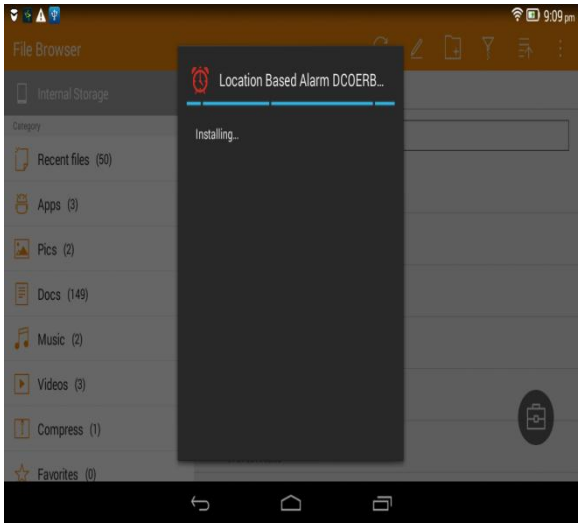


Fig. 4(b). Installing App

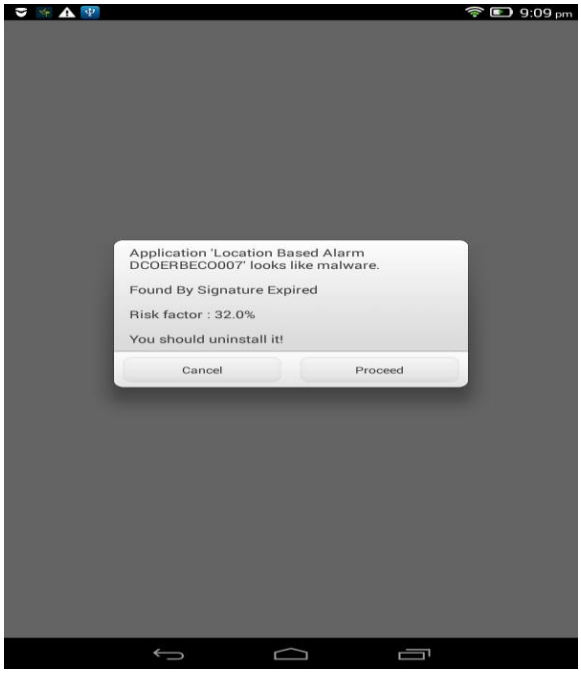


Fig.4(c). Risk score

3. CONCLUSION

Extraction of lists of permission and checking these permissions for the detection of malicious is implemented. This extraction contains numbers of permissions related to network, communication, games, personal data etc. Permissions are considered for generating risk score function resulting in user friendly risk communication. User will be notified regarding the risk of the application before installing it, consequently preventing user from installing malicious applications.

4. FUTURE WORK

Application monitoring for the system can keep an eye on applications which may contain malicious permission. Monitoring can improve the projects efficiency in terms of functionality. This monitoring of application will check the flow of data and permission used by the apps

7. REFERENCES

[1]. B.P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, "AndroidPermissions: A Perspective Combining Risks and Benets," Proc. 17th ACM Symp. AccessControl Models and Technologies (SACMAT '12), 2012.

[2]. M. Nauman, S. Khan, and X. Zhang, "Apex: Extending Android Permission Modeland Enforcement with User-Dened Runtime Constraints," Proc. Fifth ACM Symp.Information, Computer and Comm. Security, pp. 328-332, 2010.

[3]. R. Stevens, C. Gibler, J. Crussell, J. Erickson, and H. Chen, "Investigating User Privacyin Android Ad Libraries," Proc. IEEE Mobile Security Technologies (MoST '12),2012.

[4]. T. Vidas, N. Christin, and L.F. Cranor, "Curbing Android Permission Creep," Proc.Workshop Web 2.0 Security and Privacy, vol. 2, 2011.

[5]. A.P. Felt, K. Greenwood, and D. Wagner, "The Effectiveness of ApplicationPermissions," Proc.Second USENIX Conf. Web Application Development, (WebApps '11),2011.

[6]. W. Enck, P. Gilbert, B. Chun, L.P. Cox, J. Jung, P. McDaniel, and A.N Sheth,"TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoringon Smartphones," Proc. Ninth USENIX Conf. Operating Systems Design andImplementation, article 1-6, 2010.

[7]. M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "RiskRanker: Scalable and AccurateZero-Day Android Malware Detection," Proc. 10th Int'l Conf. Mobile Systems,Applications, and Services, (MobiSys '12), pp. 281-294, 2012.

[8]. D. Barrera, H.G. Kayacik, P.C. van Oorschot, and A. Somayaji, "A Methodologyfor Empirical Analysis of Permission-Based Security Models and Its Application toAndroid,"Proc. 17th ACM Conf. Computer and Comm. Security, pp. 73-84, 2010.

[9]. Christopher S. Gates, Jing Chen, Ninghui Li, Senior Member, IEEE, and Robert W.Proctor, "Effective Risk Communication for Android Apps," IEEE Transactions onDependable and Secure Computing, vol. 11, no. 3, 2014.

[10]. A.P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android Permissions:User Attention, Comprehension, and Behavior," Proc. Eighth Symp. Usable Privacy andSecurity, article 3, 2012.

[11]. https//androidobservatory.org/stats