Quantitative Security Risk Assessment of Android Permissions and Applications

Yang Wang[1], Jun Zheng[1], Chen Sun[1], and Srinivas Mukkamala[2,3]

[1] Department of Computer Science and Engineering,
New Mexico Institute of Mining and Technology, Socorro, NM 87801
{yangwang,zheng,chen}@nmt.edu
[2] The Institute for Complex Additive Systems Analysis,
New Mexico Institute of Mining and Technology, Socorro, NM 87801
srinivas@cs.nmt.edu
[3] CAaNES, LLC, Albuquerque, NM 87109

Abstract. The booming of the Android platform in recent years has attracted the attention of malware developers. However, the permissions-based model used in Android system to prevent the spread of malware, has shown to be ineffective. In this paper, we propose DroidRisk, a framework for quantitative security risk assessment of both Android permissions and applications (apps) based on permission request patterns from benign apps and malware, which aims to improve the efficiency of Android permission system. Two data sets with 27,274 benign apps from Google Play and 1,260 Android malware samples were used to evaluate the effectiveness of DroidRisk. The results demonstrate that DroidRisk can generate more reliable risk signal for warning the potential malicious activities compared with existing methods. We show that DroidRisk can also be used to alleviate the overprivilege problem and improve the user attention to the risks of Android permissions and apps.

Keywords: Android App, Android Permission, Malware, Risk Assessment, DroidRisk.

1 Introduction

The use of touchscreen based mobile devices like smartphones has seen unprecedented growth in recent years due to their portability and real-time information access. According to a recent study from research firm Ovum [10], smartphones will dominate the mobile phone market with a compound annual growth rate of 24.9% from 2011 to 2017 and 1.7 billion devices are estimated to be shipped by 2017. The dominance of smartphones is largely attributed to the Google Android mobile OS which took 68% of the global market in the second quarter of 2012 [3]. This is mainly due to it being open source and the large collection of mobile applications (apps) in the unrestricted official and third-party Android app markets. In September 2012, the Google's official app store, Google Play (previously known as Android Market [6]), hit total 25 billion app downloads with more than 675,000 apps [7].

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 226–241, 2013.

However, the popularity of Android platform has also attracted the attention of malware developers. It was reported that more than 260,000 Android devices were affected by a mobile virus called `DroidDream` in the official Android Market within 48 hours in 2011 [5]. Felt et al. did a survey of mobile malware on three different mobile platform including Android [20]. They found that the most common malicious activities are collecting user information and sending premium-rate SMS message. Zhou et al. collected 204,040 apps from five different Android markets between May to June in 2011 [27]. 211 malware were found in the collected apps, where 32 from the official Android Market and 179 from the third-party markets.

The security mechanism used in Android platform to prevent the spread of malware is the permission-based model, which protects the access to sensitive privacy data (contact list, emails, phone call logs, location etc.) and system resources (GPS, camera, WiFi etc.). An Android app needs the user's approval of the requested permissions to be installed in the user's device. Felt et al. conducted two usability studies on the effectiveness of Android permission system [21]: one Internet survey of 308 Android users and a laboratory survey of 25 Android users. The studies showed that only 17% of participants paid attention to permissions during app installation, which indicates that the current Android permission system fails to protect most users from malware.

Several recommendations were made in [21] to improve the low attention and comprehension rate of Android permission system such as re-organizing and re-naming categories, description of permissions focusing on risk instead of resources, smaller permission list etc.. However, based on our own experiences of installing Android apps, the main reason causing the ineffectiveness problem is the text-based permission warning interface for app installation, which can be easily ignored by the users [24]. Our solution for this problem is to have a quantitative assessment of the risk levels of Android permissions and apps. With the quantitative risk information, users can easily understand the risk of an app to be malicious and pay more attention to those permissions with high risk levels.

This paper has made the following contributions:

- We propose DroidRisk, a framework for quantitative security risk assessment of Android permissions and apps based on permission request patterns, which follows the U.S. National Institute of Standards and Technology (NIST) guide for IT security risk management [11]. To the best of our knowledge, this is the first attempt to quantitatively assess the risk levels of both Android permissions and apps.
- We evaluate the effectiveness of DroidRisk with two datasets. The benign app dataset has 27,274 popular apps collected from Google Play in July 2012. The malware dataset consists of 1,260 Android malware samples from the Android Malware Genome Project [26]. We show that reliable risk signal can be generated with the quantitative risk levels of apps for warning the potential malicious activities.

- We show that DroidRisk can be applied to alleviate the overprivilege problem.
- We have implemented two applications of DroidRisk to improve the user attention to the risks of Android permissions and apps: a modified App web page in Android market with the quantitative risk information of the app and its requested permissions, and an Android app to evaluate the risk levels of apps installed in local device.

2 Related Work

Due to its importance for Android security and user privacy, the permission system has attracted lots of research interests. There are several studies on how the permissions are used by Android apps. In [13], Barrera et al. did an empirical analysis of the permission-based security models by analyzing 1,100 most popular Android apps using the Self-Organizing Map (SOM) algorithm. They found that among the defined permissions only a small portion of them are actively used by developers. Another finding of their study is that the requested permissions are not strongly correlated with application categories. Felt et al. did a survey of 100 paid and 856 free apps from Android Market in [20]. It was observed that 93% of free and 82% of paid apps request at least one dangerous permission. They also built a tool called Stowaway that can detect whether a compiled Android app requests more permissions than necessary, i.e. overprivileged [18]. Among the apps they investigated, about one-third were actually overprivileged. In [25], Wei et al. studied the permission evolution in the Android ecosystem. One of their key observations is that the set of dangerous-level permissions always outnumbers other permission types in all versions of the Android platform and it is still growing. Frank et al. studied the permission request patterns of Android apps using pattern mining technique [22]. They tried to relate the permission request pattern with the app's reputation which can be served as an indicator of app quality. Although all these works revealed something about the permission request patterns of Android apps, they didn't attempt to identify potential malware.

Recently, the app's permission request pattern has been used to generate risk signal for warning potential malicious activities. Enck et al. proposed a light weight application certification service called Kirin that uses a rule-based strategy to identify suspicious apps based on their requested permissions [16]. However, because the rules were defined manually, they can't adapt to the changing characteristics of current permissions and apps. For example, the 9th rule of Kirin is no longer valid because the permission `SET_PREFERRED_APPLICATION` has been deprecated since Android API level 7. In [27], Zhou et al. proposed a system called DroidRanger to detect malicious apps in official and alternative Android markets. The first component in DroidRanger is *permission-based filtering* which uses some dangerous permissions such as `RECEIVE_SMS` and `SEND_SMS` to find potential malicious apps. It was shown that only 0.66% of apps needed further analysis after the permission-based filtering step. Chia et al. studied permissions systems of Facebook apps, Chrome extensions and Android apps to

find the reliable signals to identify potential harmful and inappropriate apps [15]. They investigated several signals including the adjusted community rating, the availability of the developer's website and the number of apps published by the developer. However, none of those signals was found to be reliable. In [24], Sarma et al. proposed a set of risk signals by examining the permission request patterns from apps in Android Market and the collected malicious apps. The proposed risk signals include rare critical permissions (RCP), rare pairs of critical permissions (RPCP), combination of RCP and RPCP, and category-based RCP (CRCP). The RCP signal is triggered if at least one of the critical permissions is requested by less than certain percentage of the Android Market apps. The RPCP signal is triggered if for a pair of critical permissions, any individual permission occurs more frequent than they occur as a pair. The CRCP signal is the combination of category information with RCP. Though RCP has shown superior performance compared with Kirin in terms of warning and detection rates [24], the users do not have any idea about the risk levels of requested permissions by an app and the app itself as there is no quantitative assessment of the risk levels.

3 Application Dataset

In this section, we describe the benign app and malware datasets collected for our study. We also provide the statistics of requested permissions of these two datasets.

3.1 Data Collection

Benign App Dataset. Among the large number of Android markets available worldwide, Google Play is the largest and most reliable one. An antivirus system, called Google Bouncer, is deployed to detect the malicious apps uploaded by developers [1]. Any malicious app found by Google Bouncer, that may be harmful to users or tries to steal privacy information, will be removed from Google Play. Thus, it is reasonable to assume that the apps from Google Play perform no malicious activities and can be used to construct the benign app dataset.

To collect the information of apps from Google Play, we developed a crawler to automatically extract the name, category and the requested permissions of each app from its corresponding web page. In total we collected the information of $27,274$ popular apps from Google Play in the middle of July, 2012. The collected apps belong to 26 subcategories under "Applications" category and 8 subcategories under "Games" category.

Malware Dataset. The Android malware dataset used in our study is from the Andriod Malware Genome Project [26]. This dataset consists of $1,260$ Android malware samples in 49 families from different markets which has a much larger size compared with the malware dataset of 121 samples used in [24]. 86% of the samples are repackaged versions of normal apps. 36.7% of them leverage

root-level exploits. More than 90% try to remote control the device. 45.3% and 51.1% of them stealthily send short messages or make calls and collect privacy user information, respectively. Unfortunately, there is no category information for this malware dataset.

3.2 Statistics of Requested Permissions

Many malware request more permissions than the need of their claimed functions to perform malicious activities. For example, a game-like malware may request the permission to send short messages, which could result in a financial loss to the user. In this section, we provide the statistics of permissions requested by benign apps and malware, which inspire our work of DroidRisk.

Frequently Requested Permissions. Figure 1 shows the top 20 requested permissions in the benign app and malware datasets. `INTERNET` is the most frequently used permission by both the benign apps and malware. There are many reasons to request permission for internet access: some of the apps need to log in; some are designed to use internet like browsers and email clients; some need to load advertisement etc. As a result, Internet-related permissions, such as `ACCESS_NETWORK_STATE` and `ACCESS_WIFI_STATE`, become very popular. Another set of widely used permissions are location related ones such as `ACCESS_FINE_LOCATION` and `ACCESS_COARSE_LOCATION` for location based services. The most significant differences between benign apps and malware observed from Figure 1 are: malware are more favor of changing the settings and use money-related services such as short message service (SMS). Changing settings, especially changing the network settings, generally is the first step before a malware performs any malicious activity. Sometimes malware even try to kill background processes, which could help them avoid being detected by anti-virus apps. It can be seen that SMS is a popular service among malware as four SMS-realted permissions are much more popular in malware than benign apps. In [26], Zhou and Jiang introduced a family of malware targeting on the financial charging. This kind of malware may stealthily edit and send out SMS to waste the user's money. They may subscribe premium-rate services without the user's consent for profit.

Number of Requested Permissions. Figures 2(a) and 2(b) show the percentages of malware and benign apps requesting certain number of permissions, respectively. As can be seen, malware are likely to request more permissions than benign apps. It is shown that 58.8% and 92.7% of the benign apps request no more than 4 and 11 permissions respectively, while the numbers for malware are 7.5% and 49.3%. Figures 2(c) and 2(d) show the percentages of malware and benign apps requesting certain number of dangerous permissions, respectively. It can be easily seen that malware also request more dangerous permission than benign apps. More than half of the benign apps request 2 or less dangerous permissions and 91.8% of them have no more than 7 dangerous permissions. For

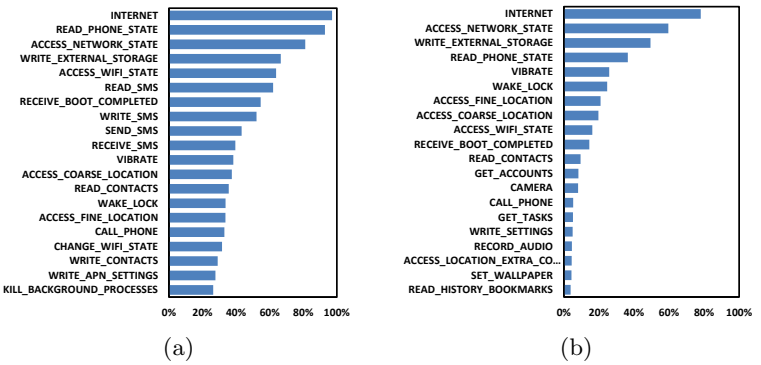

Fig. 1. Top 20 most popular permissions in: (a) malware dataset (b) benign app dataset

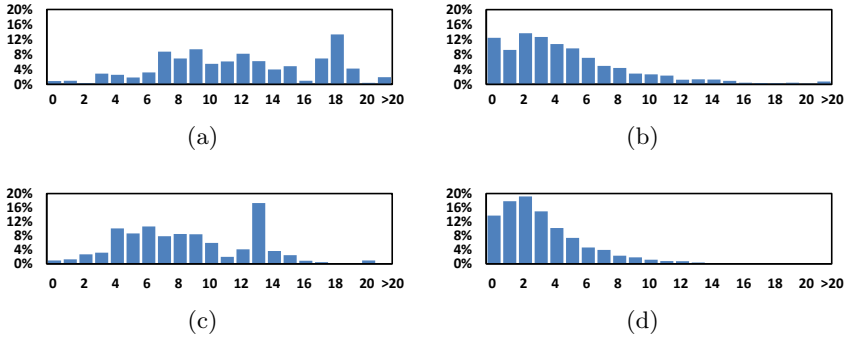

Fig. 2. Percentage of apps requesting certain number of permissions: (a) malware (b) benign apps; Percentage of apps requesting certain number of dangerous permissions: (c) malware, (d) benign apps

malware, 62.5% of them have at least 7 dangerous permissions, and around a quarter of them request 13 or more dangerous permissions. Clearly, malware are more interested in dangerous permissions for their malicious activities.

4 Methods

4.1 Quantitative Security Risk Assessment

Quantitative methods have been used to assess the financial risk for a long time [12] but they are still relatively new for security risk assessment. To guide the risk management for IT systems, NIST published a set of IT security risk management best practices in 2002 [11]. According to the NIST guide, risk assessment is the first step of IT risk management. To quantitatively assess the risk level

of an system, it needs to find the risk of each potential adverse event for the system, which is defined as [11,23]:

$$R(E_i) = L(E_i) \times I(E_i) \tag{1}$$

where E_i is the ith potential adverse event, $R(E_i)$ is the risk of E_i, $L(E_i)$ and $I(E_i)$ are the likelihood and the impact of E_i, respectively. The likelihood represents the probability that a weakness is exploited by the attacker, while the impact refers to the magnitude of harm caused by this weakness being exploited [23]. Assume the adverse events are independent, the system risk R_{sys} can be obtained by summing the risk values of individual adverse events as shown in Equation (2):

$$R_{sys} = \sum_i R(E_i) = \sum_i L(E_i) \times I(E_i) \tag{2}$$

where $i = 1, 2, ..., n$ and n is the total number of potential adverse events.

4.2 DroidRisk – Quantitative Security Risk Assessment of Android Permissions and Apps

The DroidRisk framework for quantitative risk assessment of Android permissions and apps follows the NIST guide. We consider an Android app A as the system and each permission p_i requested by A as the individual adverse event. By assuming that the permissions requested by A are independent, the risk level of the app A, R_A, can be defined as:

$$R_A = \sum_i R(p_i) = \sum_i L(p_i) \times I(p_i) \tag{3}$$

where $R(p_i)$ is the risk level of permission p_i, $L(p_i)$ and $I(p_i)$ are the likelihood and the impact of permission p_i respectively, $i = 1, 2, ..., n$ and n is the total number of requested permissions by A.

The key problem to be solved in the DroidRisk framework is to calculate the likelihood $L(p_i)$ and the impact $I(p_i)$ for a requested permission p_i. We define the likelihood $L(p_i)$ as the probability that the app A is malware if p_i is requested by A, i.e. $P(A \text{ is malware} \mid p_i)$. This *posteriori* conditional probability can be calculated using Bayes' rule as show in Equation (4):

$$P(A \text{ is malware} \mid p_i) = \frac{P(p_i \mid A \text{ is malware}) \times P(A \text{ is malware})}{P(p_i)} \tag{4}$$

where $P(p_i \mid A \text{ is malware})$ is the priori probability that a malware requests permission p_i, $P(A \text{ is malware})$ is the priori probability that an app is malware for all collected apps, and $P(p_i)$ is the priori probability that any collected app, benign or malicious, requests permission p_i.

For estimating impact levels of permissions, we consider two classes of permissions, normal and dangerous[1]. Although it is hard to evaluate the exact level of harm caused by a permission if it is requested by a malware, it is certain that dangerous permissions are more harmful than normal permissions. Thus, we set the impact level of normal permissions, I_{np} as 1 while give dangerous permissions a higher impact level. In Section 5.2, we used an empirical method to determine the impact level of dangerous permissions, I_{dp}.

5 Results

In this section, we first present the likelihood values calculated for Android permissions based on two datasets we collected. We then show how to determine the impact level for dangerous permissions with an empirical method followed by the presenting of risk levels of Android permissions and apps from the two datasets. Finally we demonstrate that DroidRisk can be used to assess the risks of apps in third-party markets.

5.1 Likelihood of Android Permissions

With the collected benign app dataset and malware dataset, we can calculate the likelihood of each Android permission using Equation (4). Figure 3 shows the top 20 permissions with highest likelihood values. The blue bar represents the likelihood of a normal permission while the red bar is for the likelihood of a dangerous permission.

There are 14 dangerous permissions on the list. `WRITE_APN_SETTINGS` has the highest likelihood among all permissions, which can change the network setting, thus to intercept and inspect the network traffic without the user's awareness. There are 5 SMS and MMS related dangerous permissions on the list, which means that SMS and MMS services are the major target of malware developers. Unlike other systems which don't pay attention to normal class permissions, we have 6 of them on the list. These permissions may be used by malware to facilitate some malicious activities. For example, `INSTALL_PACKAGES` and `DELETE_PACKAGES` are used by several malware families such as `JSMSHider` and `GoldDream` to perform update attack and `KILL_BACKGROUND_PROCESSES` is used by an Android Trojan named `AnserverBot` to avoid being detected by certain anti-virus apps [26][2].

[1] Android characterizes the potential risk of the permissions using 4 protection levels – normal, dangerous, signature and signatureOrSystem. Since signature or signatureOrSystem permissions can't be granted to third party apps, we category them into normal class. The protection level for a given permission in this study is obtained from Android API level 9.

[2] `KILL_BACKGROUND_PROCESSES` has been used to replace the old name `RESTART_PACKAGES` for permission - 'kill background processes' since the release of Android API level 8.

5.2 Impact Level of Dangerous Permissions

Based on the discussion in Section 4.2, the impact level of dangerous permissions, I_{dp} should be higher than the impact level of normal permissions, I_{np}. Since there is no way to obtain the actual level of harm caused by normal permissions or dangerous permissions, we solved the problem with an empirical method using ROC (Receiver Operating Characteristic) curve. ROC curve is an efficient tool to evaluate the binary classification performance and select the optimal threshold setting [17]. In our case, once I_{dp} is determined, we can obtain the risk level of each Android permission, and then compute the risk levels of benign apps and malware according to their requested permissions. As one of our design goals is to classify benign apps and malware with a single threshold of risk level, ROC curve is the appropriate choice to find the value of I_{dp} which gives the best classification performance.

ROC curve is plotted with the true positive rate (TPR) versus the false positive rate (FPR) under various threshold settings. Figure 4 shows the ROC curve generated for $I_{dp} = 2$. In Figure 4, the ideal classification performance is obtained at the upper left corner, which means no classification error. The closer the ROC curve to the upper left corner, the better the classification performance. Therefore, a popular metric used to measure the classification performance is the area under curve (AUC) [14], which is the area between the ROC curve and the x-axis. Usually a AUC greater than 0.9 is considered as excellent classification performance.The value of I_{dp} is determined as 1.5 which gives the highest AUC value (0.9313).

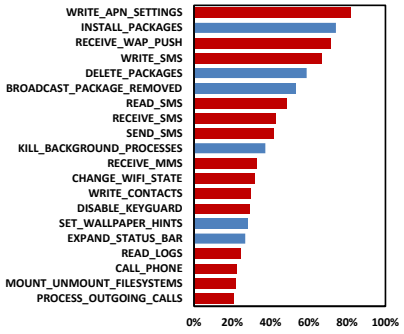

Fig. 3. Top 20 permissions with highest likelihood value

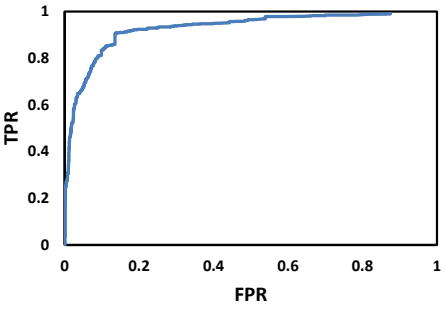

Fig. 4. ROC curve with $I_{dp} = 2$

5.3 Risk Levels of Android Permissions and Apps

Once the impact value of dangerous permissions is obtained, we are able to compute the risk levels of all Android permissions. Figure 5 shows the top 20 permissions with highest risk levels. Compared with the top 20 list for likelihood, there is no new permission on the list but the rankings of most dangerous

permissions are promoted because of higher impact level. `WRITE_APN_SETTINGS` is still on the top of the list with highest risk level among all permissions.

We also compute the risk levels for all benign apps and malware in our datasets by simply summing the risk levels of requested permissions as shown in Equation (3). Figure 6 shows the histograms of the risk levels of benign apps and malware. It can be easily observed that majority of benign apps have risk levels less than 2, while malware typically have higher risk levels.

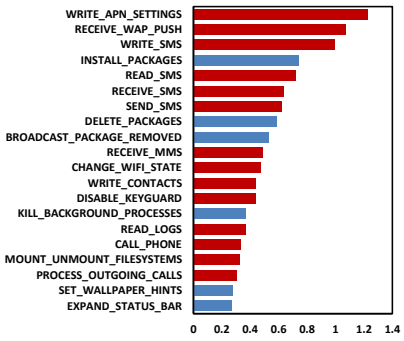

Fig. 5. Top 20 permissions with highest risk levels

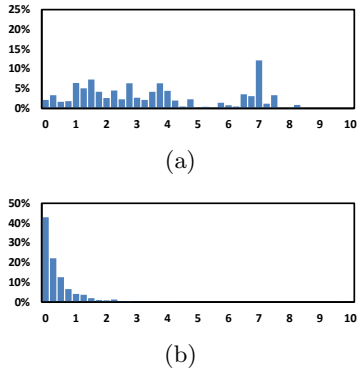

Fig. 6. Risk level histograms, (a) malware, (b) benign apps

As seen in Figure 6(b), there are few benign apps with very high risk levels. We did an investigation on top 20 benign apps with highest risk levels which are shown in Figure 7. The top 6 apps on this list are solely used for in-store demonstration of certain mobile devices. Some of them are warned by the vendors not for public use. The possible reason is that such apps may need lots of permissions to demonstrate different features of the device. It's not surprising to see 6 security apps on this list since they usually require lots of permissions to monitor the whole system and may block, intercept or change the behaviors of the system or other apps. Therefore, it is highly recommended that users should carefully choose the security apps for their devices since there are malware pretending to be anti-virus apps [2]. The rest of the apps on this list are communication tools used for SMS, calls, email exchange etc. Since SMS, calls and other communications may leak privacy information or result in unexpected financial charge, users need to pay special attention to those apps.

5.4 Third Party Android Market

Besides Google Play, there are lots of third party android markets available worldwide, which are considered as the major source of malware [27]. Many of such markets are not capable to do a malware detection before they put a new app on the shelf. Some of them even solely depend on the feedback from

users to decide whether an app needs to be removed from their markets or not. However, third party markets are more preferred than Google Play by many users. One reason is that these users need free apps which cannot be found in Google Play, e.g. free app of the day in Amazon appstore for Android. Another reason is that some users need a localized version of the original app, especially for users whose native language is not English. We arbitrarily selected a third party Android market in China, mumayi [9], to investigate the risk of third party Android markets.

We downloaded 602 popular apps from mumayi using a crawler and collected their requested permission information. We then computed the risk level of each collected app using DroidRisk. Figure 8 shows the boxplots of the risk levels of apps from Google Play, apps from mumayi and malware. The median risk levels for apps from Google Play, apps from mumayi and malware are 0.32, 1.35 and 3.22, respectively. This indicates that to install an app from the third party market is more risky than the official market, Google Play.

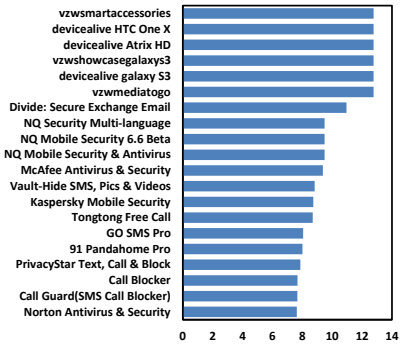

Fig. 7. Top 20 benign apps with highest risk levels

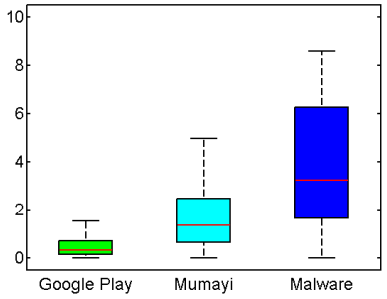

Fig. 8. Boxplots of the risk levels of apps from Google Play, apps from mumayi and malware

6 Applications of DroidRisk

In this section, we show that reliable risk signal to identify potential malware can be generated with the quantitative risk information obtained by DroidRisk. The applications of DriodRisk for alleviating the overprivilege problem and improving the user attention to the risks of Android permissions and apps are also presented.

6.1 Reliable Risk Signal

Risk signals are used by users to identify potential malware. A reliable risk signal should be triggered by as many malware as possible, and rarely be triggered by

benign apps. We compare the performance of DroidRisk with two state-of-art methods: Kirin from [16], RCP and $RCP + RPCP$ from [24] using 10-fold cross-validation. The risk signal for RCP is generated with rule $\#RCP(\theta) \geq 1$ which is the simplest one in [24]. The risk signal for $RCP + RPCP$ is generated with rule $\#RCP(2) + \#RPCP(1) \geq \theta$ which is the best performing one in [24]. The performance of Kirin is obtained with 7 rules as described in [24]. Figure 9 presents the ROC curves of different method for fold-1. For other folds, we obtained similar results. From Figure 9, we can observe that DroidRisk has significant better classification performance than other methods. One possible reason is that DroidRisk tries to capture the request patterns of both normal and dangerous permissions while there are only 9 manually defined rules in Kirin and only 24 or 26 critical permissions are used for RCP and $RCP + RPCP$.

Figure 9 also shows that the risk signal of Kirin is not tunable as the rules are predefined. Although the risk signals generated by RCP and $RCP + RPCP$ are adjustable, it is not easy to find a reliable one since there are several parameters to be tuned. With the quantitative risk information generated by DroidRisk, the risk signal is generated from a single threshold of risk level which can be easily found using ROC curve. The simple thresholding operation also makes it suitable to be implemented in resource-constrained mobile devices.

The results of the 10-fold cross validation are shown in Figure 10, where F-score is used as the performance measure. The parameters of DroidRisk, RCP and $RCP + RPCP$ are chosen as the best performing ones for the training sets. It can be seen that DroidRisk has the best performance among all the methods.

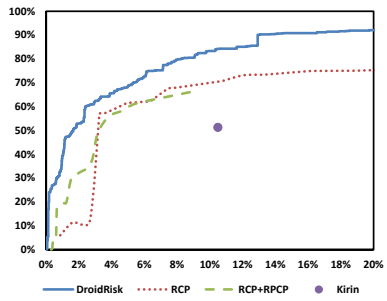

Fig. 9. ROC curves of different methods for fold-1

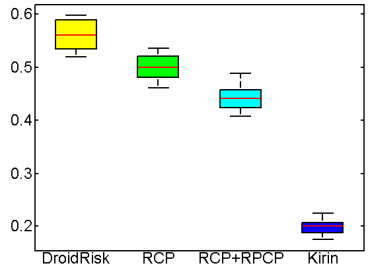

Fig. 10. Boxplots of the F-scores of different methods for 10-fold cross validation

6.2 Overprivilege Problem

Android permission system gives the app developers the ability of requesting any permission no matter they actually use it or not. Many developers and users rarely pay attention to whether the requested permissions are useful for them or not. As a result, Android apps tend to be overprivileged, which may raise security problems. We randomly selected 50 popular apps from both the third

party Android market, mumayi, and Google Play. We used the Stowaway [18] to detect unnecessary permissions in each app. Table 1 shows the statistics of the collected apps due to the overprivilege problem. The boxplots in Figure 11 show how the unnecessary permission affect the risk levels of apps from mumayi and Google play. It can be seen that the overprivilege problem is common in both the official and third party markets although the problem is much severe in third party market. The median risk level of apps from mumayi has increased from 1.04 to 1.43 due to overprivilege. Although the median risk level of apps from Google Play only increases from 0.64 to 0.66, we did find 7 apps whose risk levels raise above the threshold set by DroidRisk because of those unnecessary permissions, which makes them suspicious to be malware. Since the app developers always try to get more downloads, declaring unnecessary permissions to result in a high risk level for the app will make the user more likely to cancel the download because of the warning of our DroidRisk system. Thus, the overprivilege problem can be alleviated since the developers need to be more careful about requesting permissions for their apps, especially those related to possible malicious activities.

Table 1. Statistics of the collected apps from mumayi and Google Play due to overprivilege problem

	mumayi	Google Play
Percentage of overprivileged apps	64%	44%
Average # of unnecessary permissions	1.92	0.84

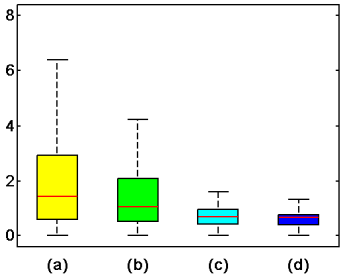

Fig. 11. Risk levels with and without unnecessary permissions for apps from, (a-b): mumayi, (c-d): Google Play

6.3 Improving User Attention to Risks of Android Permissions and Apps

One problem with current Android permission system is that all information related to the risks of permissions are text descriptions while users often do not pay attention to the text information due to various reasons [21]. One of the design goals of DroidRisk is to improve the user attention of the risks of Android permissions and apps. In the following, we show two applications that utilize the quantitative risk information obtained by DroidRisk to reach this design goal.

App Web Page with Quantitative Risk Information: Each Android app has a web page after it is submitted to an Android market, official or third party. Users often browse the market to find out interesting apps for their devices.

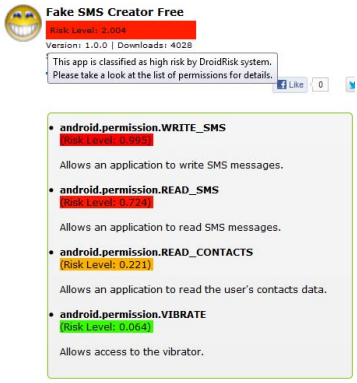

Fig. 12. Screenshot of the Fake SMS Creator Free app's web page in Aptoide with quantitative risk information

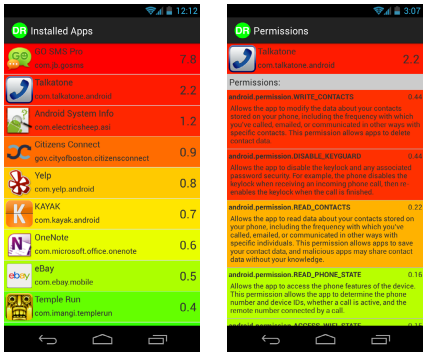

Fig. 13. DroidRisk Android App: (a) view of the list of installed apps with their corresponding risk levels, (b) view of the list of permissions requested by an app and their risk levels

Since the text-based permission information shown in the app's web page is not effective in informing the user about the risk of the app, we developed a solution to present the quantitative risk information of the app and its requested permissions in the web page. The solution was implemented in FireFox browser by creating a GreaseMonkey[3] script to change the display of the app's web page. Figure 12 uses the Fake SMS Creator Free app's web page in Aptoide, a third party market[4], as an example to show how the risk information of the app and its requested permissions are displayed in the web page. The current design uses a colored box that displays the risk value of an app or a requested permission. The color scheme of the design corresponds to the risk level. When the user's mouse moves above the box of the app's risk level, a textbox will be shown to alert the potential risk and ask the user to see the list of permissions. It is expected that this design with the quantitative risk information and the color-based visual cue can significantly improve the attention rate of Android permission system.

DroidRisk Android App: To let the users evaluate the risks of apps installed in their local devices, we developed an Android App of the same name as the framework. This app extracts the permissions requested by each installed app and computes the risk level of the app. Figure 13(a) shows the app's interface which shows the list of installed apps with their corresponding risk levels. The user can tap any app to see the list of permissions requested by the app and their risk levels (Figure 13(b)). The app also uses a color scheme to give the users a direct visual cue of the risk level. We plan to expand the functionalities of DroidRisk app in future to report the risk level of any app in any Android market once the user provides the corresponding information.

[3] GreaseMonkey is a Firefox add-on that allows the user to customize the display or behavior of a web page using a user script [8].

7 Conclusion

In this paper, we demonstrate that the proposed DroidRisk framework can be used to improve the efficiency of Android permission system for informing the user about the risks of Android permissions and apps. It can be easily incorporated into existing Android malware detection systems as the first barrier to prevent the spread of malware. It will be especially useful for those third party markets without malware detection capability. In our future work, we will investigate alternative ways to find the impact levels of Android permissions, e.g. based on the study of [19] which ranks the risks of permissions according to the users' ratings. We will also explore security rules involving multiple permissions and evaluate their corresponding risks which may be underestimated by DroidRisk.

References

1. Android and security, http://googlemobile.blogspot.com/2012/02/android-and-security.html
2. Android enesoluty - fake antivirus, spyware, http://contagiominidump.blogspot.com/2012/11/android-enesoluty-fake-antivirus-spyware.html
3. Android races past apple in smartphone market share, http://money.cnn.com/2012/08/08/technology/smartphone-market-share/index.html
4. Aptoide, http://www.aptoide.com
5. Drioddream malware has now claimed 260,000 devices, http://androidjournalist.wordpress.com/2011/03/10/droiddream-malware-has-now-claimed-260000-devices/
6. Google play, https://play.google.com
7. Google play hits 25 billion downloads, http://officialandroid.blogspot.com/2012/09/google-play-hits-25-billion-downloads.html
8. Greasemonkey, http://www.greasespot.net
9. Mumayi, http://www.mumayi.com/
10. Smartphone sales to top 1.7 billion by 2017, dominate mobile phone market, http://www.inquisitr.com/230416/smartphone-sales-to-top-1-7-billion-by-2017-dominate-mobile-phone-market/
11. National Institue of Standards and Technology (NIST), Risk management guide for information technology systems (2002)
12. Alexander, C.: Market risk analysis: quantitative methods in finance. Wiley (2008)
13. Barrera, D., Kayacik, H.G., van Oorschot, P.C., Somayaji, A.: A methodology for empirical analysis of permission-based security models and its application to android. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, pp. 73–84 (2010)
14. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recogn. 30(7), 1145–1159 (1997)

15. Chia, P.H., Yamamoto, Y., Asokan, N.: Is this app safe?: a large scale study on application permissions and risk signals. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012, pp. 311–320 (2012)
16. Enck, W., Ongtang, M., McDaniel, P.: On lightweight mobile phone application certification. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS 2009, pp. 235–245 (2009)
17. Fawcett, T.: An introduction to roc analysis. Pattern Recognition Letters 27(8), 861–874 (2006)
18. Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D.: Android permissions demystified. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, pp. 627–638 (2011)
19. Felt, A.P., Egelman, S., Wagner, D.: I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In: Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM 2012, pp. 33–44 (2012)
20. Felt, A.P., Greenwood, K., Wagner, D.: The effectiveness of application permissions. In: Proceedings of the 2nd USENIX Conference on Web Application Development, WebApps 2011, p. 7 (2011)
21. Felt, A.P., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D.: Android permissions: user attention, comprehension, and behavior. In: Proceedings of the Eighth Symposium on Usable Privacy and Security, SOUPS 2012, pp. 3:1–3:14 (2012)
22. Frank, M., Dong, B., Felt, A.P., Song, D.: Mining permission request patterns from android and facebook applications. In: Proceedings of the IEEE International Conference on Data Mining, ICDM 2012 (2012)
23. Joh, H., Malaiya, Y.K.: Defining and assessing quantitative security risk measures using vulnerability lifecycle and cvss metrics. In: Proceedings of the 2011 International Conference on Security and Management, SAM 2011, pp. 10–16 (2011)
24. Sarma, B.P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., Molloy, I.: Android permissions: a perspective combining risks and benefits. In: Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, SACMAT 2012, pp. 13–22 (2012)
25. Wei, X., Gomez, L., Neamtiu, L., Faloutsos, M.: Permission evolution in the android ecosystem. In: Proceedings of the 2012 Annual Computer Security Applications Conference, ACSAC 2012 (2012)
26. Zhou, Y., Jiang, X.: Dissecting android malware: characterization and evolution. In: Proceedings of the 33rd IEEE Symposium on Security and Privacy, Oakland 2012, pp. 95–109 (2012)
27. Zhou, Y., Wang, Z., Zhou, W., Jiang, X.: Hey, you, get off my market: detecting malicious apps in official and alternative android markets. In: Proceedings of the 19th Network and Distributed System Security Symposium, NDSS 2012 (2012)