Permission-Based Abnormal Application Detection for Android

Jiawei Zhu[1,2,3], Zhi Guan[1,2,3,⋆], Yang Yang[1,2,3], Liangwen Yu[1,2,3], Huiping Sun[1,2,3], and Zhong Chen[1,2,3]

[1] Institute of Software, School of EECS, Peking University, China
[2] MoE Key Lab of High Confidence Software Technologies (PKU)
[3] MoE Key Lab of Network and Software Security Assurance (PKU)
{zhujw,guan}@infosec.pku.edu.cn

Abstract. Android has become one of the most popular mobile operating system because of numerous applications it provides. Android Market is the official application store which allows users to search and install applications to their Android devices. However, with the increasingly number of applications, malware is also beginning to turn up in app stores. To mitigate the security problem brought by malware, we put forward a novel permission-based abnormal application detection framework which identifies potentially dangerous apps by the reliability of their permission lists. To judge the reliability of app's permissions, we make use of the relation between app's description text and its permission list. In detail, we use Naive Bayes with Multinomial Event Model algorithm to build the relation between the description and the permission list of an application. We evaluate this framework with 5,685 applications in Android Market and find it effective in identifying abnormal application in Android Market.

Keywords: Android, Abnormal Application, Permission Reliability.

1 Introduction

Nowadays, smartphones occupy an important position in people's daily life. They allow users to communicate, surf the Internet or have all kinds of entertainments at any place. The most common mobile operating systems used by modern smartphones include Android, iOS and Symbian. Android, which is an open source Linux-based mobile operating system distributed by Google, is one of the most popular mobile operating systems today. Because of its open architecture which is convenient to develop and debug the applications, more and more developers turn to pay attention to this rising system.

With the arising of numerous Android applications, there exists lots of app stores providing more convenient platforms for users to search and install the free and paid applications. Among all these Android app stores, Google Play Store [1] (named as Android Market originally) is the official app store for Android

⋆ Corresponding author.

smartphone users. A research report released by AppBrain showed that the total number of apps in Android Market was over 450,000 at the beginning of June, 2012 [2]. In Android Market, users can browse the description, application's information and the permission list in the application's main interface. For most users searching applications in the market, they usually make a decision on selecting which applications to install based on three aspects: the introductions of applications (including descriptions and application screenshots), ratings and other users' reviews.

Unlike Apple's App Store, Google has minimal involvement in Android Market. Android Market provides diverse applications not only developed by famous corporations, but also by some small companies or amateur individual developers. Once published, apps from the Android Market can only be removed by Google because of being reported malicious or their content violating terms of use. So without Google's restrict check on applications, Android Market may contain some malicious applications. For this reason, some measures should be taken to ensure the security of Android devices.

In this paper, we propose a method to analyze the potential security problem of an application in app stores. In detail, we put forward a permission-based abnormal application detection framework to mitigate the security problem in Android Market by the reliability of app's permissions. We design a detailed predicting model which is the most important part in the framework to reflect the relation between the description and the permission list of an application. With this model, we can predict the actually needed permission list of an application in Android Market based on its description.

We use 5,685 free applications in the Android Market to evaluate our permission predicting models and find that this model is effective in predicting the permission list of an application in Android Market. Besides, we apply our security framework in the detection of reliability of applications' permissions and give out a test on real malware announced by Google.

The following sections of this paper proceeds as follows. Section 2 describes the related work about Android malware detection. Section 3 overviews our framework. Section 4 describes the process of our experiment. Section 5 evaluates this model on 5,685 apps in Android Market. Section 6 applies our method on detecting the reliability of permissions and few malicious apps announced by Google and section 7 concludes.

2 Related Work

Smartphone security is a growing concern in recent year. Static analysis and dynamic analysis are two main approaches for the detection of malware. In Android platform, Enck et al. [3] present a dynamic tainting analysis to protect the security of users' sensitive data. They labeled the information with different types. At last, the system made a result based on the policies. They also tested this tool on 30 applications and found that 20 of these applications taking suspicious actions on users' data.

Enck et al. [4] also designed a tool to decompiled the Android executable file into Java source code to make a static analysis to identify malicious Android applications. They studied 1,100 Android applications with this method and obtained 27 finds including the leakage of phone identifiers, location information sent to advertisement servers and some specific attacks on Android OS.

Same authors [5] designed and implemented a framework to identify the dangerous applications based on their certain combinations of permissions. They designed Kirin which modified the application installer with this method to prevent the malicious applications.

Portokalidis et al. [6] utilize virtualization method to detect the security of Android applications. In detail, they put forward a method to analyze the security of applications in the remote servers which held the mirror of smartphone in the virtual environment. They implemented this system and took some analysis on the parameters of this system such as battery level and CPU utilization.

Zhou et al. [7] proposed a permission-based behavioral footprinting scheme and heuristics-based filtering scheme to detect the malware. The former compared the application with the known Android malware based on their permissions to detect the new sample of them. The latter was used to identify the unknown malicious families. The experiment was taken with 204,040 apps collected from 5 different app stores.

Burguera et al. [8] made use of k-means method with the time of system call as the features to identify the malware. The experiment was tested on two known Android malware.

3 Security Framework

3.1 Permission-Based Abnormal Application Detection Framework Overview

In this paper, we put forward a novel permission-based abnormal application detection framework in Android Market. The main point of our framework is that the description of an application is closely related to its permission list in Android platform. Here, the description of an application, which can be found in Android Market, describes the features and the functions of this application. In other word, the description information concretes the functions of the application implementing. Permission-based security model is one of the most important security measures of Android devices. Permission model is used to restrict to access to some special resources. It can also be said to restrict to take some potentially dangerous actions. If an application wants to accomplish some specific actions, it has to request the corresponding permissions. So the permission list of an application reflects the actions, or even functions of this application. For example, if an app defines the function of sharing files in its description text, this sharing files function should contain two-step actions. The first one is to find the users to be shared with the file from contact. The other is transferring the file to the selected users via network. Based on these two-step actions, this application should request `READ_CONTACT` and `INTERNET` permissions to realize its function. Therefore, we believe that the description and the permission list of an application are closely related. However, for the malicious applications, they usually hide some potentially dangerous actions which lead to their actions being not accord to the described functions.

Based on this intuitive conclusion, we provide a permission-based abnormal application detection framework to analyze the potential dangers of an app in Android Market. The detailed framework is shown in Figure 1. In this framework, the most crucial part is the model which reflects the relation between the description and the permission list. With this model, we can predict an app's normal permission list. Furthermore, after the analysis of the permission comparator, if the permission list of an app is not accord to its predicted normal permission list, we think this app is with hidden danger.

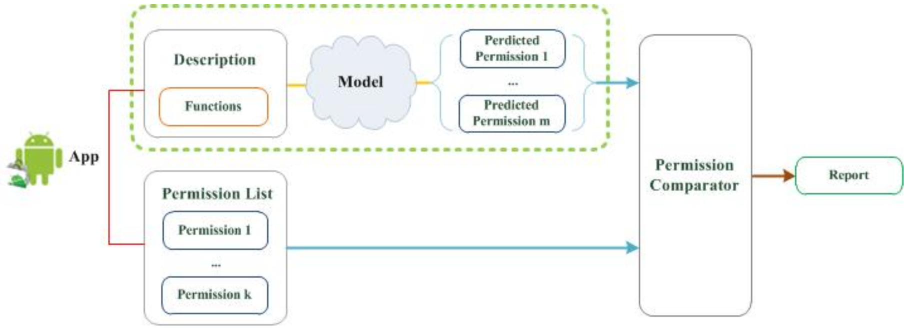

Fig. 1. permission-based abnormal application detection framework

In this platform, we can model the description and the permission list of an application to have a research on our security framework. In this concrete environment, we define the description and the permission list of an application as the following signs. Formally, let $p = [p_1, p_2, \dots p_k]^T$ be the output permission list, in which $p_k \in \{0, 1\}$ denotes whether permission indexed as k should be requested by an application. Let $d = (x_1, x_2, \dots x_{|V|})$ denotes the description of an application, in which x_i is the index of a word in the dictionary, V is the number of words in the dictionary. Thus, our work in the next several sections is to analyze the usability of our framework in Android. More specifically, we utilize this security framework to predict the hidden dangers of an application based on its description. In our paper, we only detect the reliability of the application's permission list and view this as the criterion of the hidden dangers of the application.

3.2 Model Selection

Based on our research problem, we should model the relation between the description and the permission list of an application well in Android platform. In detail, we use machine learning techniques to predict the permissions of an app based on the description of this app. Here, the description which is requested to be provided by developer of this application can be found in Android Market. The basic structure diagram is shown in Figure 2. Because we use supervised learning method to model this relationship, we should collect enough training examples at first. After learning the training examples' relation between the description and permission list, the model can be used in predicting the really needed permission list of an arbitrary application.

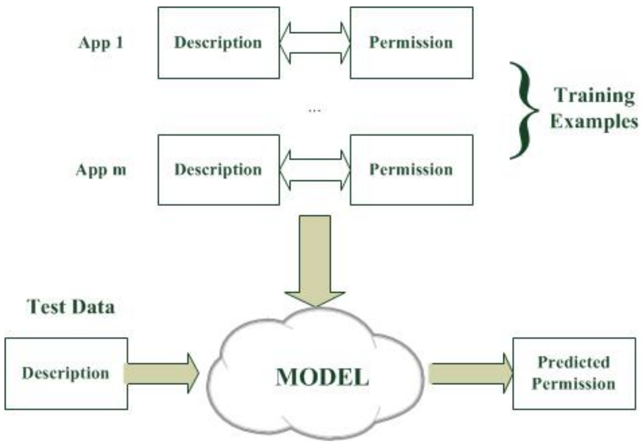

Fig. 2. Model Structure Diagram

We use Naive Bayes algorithm with multinomial event model [9] to classify the description of an application based on different permissions. This algorithm is a specialized version of Naive Bayes algorithm. Multinomial Naive Bayes algorithm models the words according to multinomial distribution. That is to say, it not only considers whether a word occurred or not, but also take the times of a word occurs into consideration. This algorithm is proved to have a better result than Naive Bayes algorithm in most occasions [10].

Formally, we have defined $p^{(i)} = [p_1^{(i)}, p_2^{(i)}, \dots p_k^{(i)}]^T$ be the i^{th} application's output permission list, in which $p_k^{(i)} \in \{0, 1\}$ denotes whether permission indexed as k should be requested by the i^{th} application. Besides, we denote $dic_1, dic_2, \dots, dic_k$ to be the dictionary of the k different permissions. let $d_k^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots x_{|V_k|}^{(i)})$ denotes the description corresponding to the i^{th} application's k^{th} permission, in which $x_j^{(i)}$ is the index of a word x_j occurring in dic_k and V_k denotes the number of words in dic_k. Multinomial Naive Bayes model will label the k^{th} permission as 0 or 1 based on the equation:

$$p_k^{(j)*} = \arg\max_{y \in p_k^{(j)}} P(p_k^{(j)}) * \prod_i P(x_i^{(j)} | p_k^{(j)})$$

At last, the model will output a predicted real permission list of an application.

4 Experiment

After overviewing our framework, we will model the relation between the description and the permission list of an application in Android platform. In our work, we aim to analyze the really needed permissions of an application from the perspective of its functions. To evaluate our model built in Section 3.3, we have an experiment on the most influential application store - Android Market - to do our experiment.

4.1 Selection of Permissions to Be Predicted

We crawled 8,050 applications (top 350 apps in 23 categories) in April, 2011 in Android Market to get the statistics of permissions. Here, we calculate the number of 122 different kinds of permissions [11] occurring in these 8,050 applications. From this statistics, `INTERNET` was the most frequently requested permission. The number of this permission occurring in 8,050 apps was nearly 7,000. That is to say, most of these apps requested this permission to access internet, which shows the spread of mobile internet. There were also other permissions requested a lot by applications, such as `ACCESS_COARSE_LOCATION` (requested 2115 times), `ACCESS_FINE_LOCATION` (requested 2127 times), `ACCESS_NETWORK_STATE` (requested 3755 times). Except these frequently occurring permissions, nearly 100 of 122 permissions occurred only few times in these apps, which agrees with the opinion in [12]. Because many of the Google defined Android permissions are not common occurring in our dataset, we omit these uncommon permissions. In this paper, we will only focus on the permissions frequently occurring.

Based on the times occurring of different permissions in our statistics, we select the permissions which are valuable to be researched. The rule we select the permissions is as follows: we will pick up the permissions based on their occurring times (we define the times as more than 200 in this paper) in these 8,050 applications. For these permissions, we can use enough training examples to give a prediction on this kind of permissions. At last, we choose 23 common permissions to make a prediction, including `ACCESS_COARSE_LOCATION`, `READ_CONTACTS` and `CALL_PHONE`.

4.2 Dataset

Because we use a supervised learning method to make a prediction on the real permission list of an application in Android Market, we have to obtain enough training examples whose permission list is trusted to do this experiment. We crawled 8,050 applications (top 350 apps in 23 categories, including their descriptions and permission list) in Android Market in April, 2011, as discussed before. Here, we assume that these apps are all well-written and not malicious because of the top rank of these apps. So we can use these high-quality apps as training and testing examples to evaluate our model. However, some of these applications' descriptions are not written in English, so we will not use these apps to train our model. Finally, we use 5,685 apps to do this experiment. The process of the experiment is as follows.

- Step 1: Put all the words occurring in applications' descriptions into the dictionary.
- Step 2: For each different permission, we use mutual information filter to pick up the words with high mutual information to this permission to generate a new dictionary which owns words have a great influence on this permission.
- Step 3: Model to predict all 23 permissions in Multinomial Naive Bayes algorithm. Using 10-fold cross validation to test the quality of the models for different permissions.

5 Results

In this section, we evaluate the model with 5,685 apps in Android Market. The prediction of permissions of these apps is made by the model of different permissions. To analyze the quality of the model, we compare the real permissions requested by the applications with the predicted permissions. We analyze the result of the prediction of permissions from 2 aspects:

1. We measure the area under the ROC curve (AUC) of different permissions' models to estimate how well our method does to build the relation between the description and the permission list of an application in this problem.
2. We pick up few words which have great influence on the prediction of corresponding permissions to analyze the influence of different words on these permissions.

5.1 Evaluation of Model

We use 10-fold cross validation to have a test on this dataset to evaluate the proposed model's quality. At first, we get the Receiver Operating Characteristic (ROC) curves and the Area Under roc Curve (AUC) [13] form the test applications. AUC is metric to evaluate the accuracy of classifying.

Then, we list the value of AUC of 23 permissions in Table 1. From this table, we find the quality of the models predicting different permissions is fine. All of these models' AUC are above 0.8. So for the models of 23 permissions, they reasonably predict the result of permission list. What's more, 12 of 23 permissions have the AUC value greater than 0.9 such as `INTERNET`, `RECEIVE_SMS`, `READ_CONTACTS`. For these excellent predicted permissions, it indicates that some certain functions of an application can obviously reflect the existence of these permissions. For example, the function of making a call or sending messages can definitely correspond to the permission about call or message. For some permissions, we guess the reason why the model on these permissions does not

Table 1. AUC value of the model on 23 permissions

Permission	AUC	Permission	AUC
ACCESS_COARSE_LOCATION	0.864	SET_WALLPAPER	0.922
ACCESS_FINE_LOCATION	0.851	WRITE_EXTERNAL_STORAGE	0.913
ACCESS_NETWORK_STATE	0.806	GET_ACCOUNTS	0.902
ACCESS_WIFI_STATE	0.845	GET_TASKS	0.853
CHANGE_WIFI_STATE	0.867	KILL_BACKGROUND_PROCESSES	0.934
INTERNET	0.904	WRITE_SETTINGS	0.910
READ_PHONE_STATE	0.880	CALL_PHONE	0.928
RECEIVE_SMS	0.924	SEND_SMS	0.964
READ_CONTACTS	0.915	WAKE_LOCK	0.877
WRITE_CONTACTS	0.944	VIBRATE	0.843
CAMERA	0.909	RECEIVE_BOOT_COMPLETED	0.872
RECORD_AUDIO	0.898		

predict so well is that some certain functions may be mapped into few possible permissions. For instance, function about connecting to the internet may relates to the internet permission or WiFi permission. We can use either of these two permissions or both of these to realize our functions. This factor may influence the accuracy of the model on these permissions to some degree. In general, this result shows that it is effective to predict the real permission list of an application based on its description. So we believe that our model is an effective model to build the relation between the description and the permission list.

5.2 Pick Up Influential Words

Next, we will pick up some influential words based on the parameters of the models of each permission. Here, we define the influential words with two features.

- The words should occur frequently in the description of applications. In this paper, we define the frequency as 100 according to 5,685 apps.
- The words should have dominant impact (the dominant impact in this paper is defined as that the positive/negative impact on the occurrence of a permission is as five times as the negative/positive impact on the occurrence of this permission) on positive or negative side to some certain permissions as well.

After selecting influential words based on these two features, we list the result of some of these words in Table 2.

From the words we list in Table 2, we find the result of these influential words is basically fit in with our common sense. For example, in the permissions about location, word "weather" is positive to the occurrence of ACCESS_COARSE_LOCATION, but is not positive to ACCESS_FINE_LOCATION. In contrast, word "GPS" is positive to ACCESS_FINE_LOCATION but not COARSE. For most applications, we think that

Table 2. Influential words according to different permissions

Permission	Positive Words	Negative Words
ACCESS_COARSE_LOCATION	location, map, weather	bible, wallpaper, word
ACCESS_FINE_LOCATION	GPS, location, map	bible, dictionary, wallpaper
ACCESS_NETWORK_STATE	ringtone	dictionary, phrases
ACCESS_WIFI_STATE	dictionary, word	calculator, jokes, ringtone
CHANGE_WIFI_STATE	Wi-Fi	book, dictionary, joke
INTERNET	news, online, vedio,	keyboard, plugin
READ_PHONE_STATE	radio, ringtone, word	locate
RECEIVE_SMS	family, message, SMS	dictionary, image, joke
READ_CONTACTS	call, contact, message	dictionary, English, game
WRITE_CONTACTS	contact, group, message	calculator, dictionary, game
CAMERA	photo, picture, camera	sound, joke, dictionary
RECORD_AUDIO	call, record, voice	wallpaper, game, weather
SET_WALLPAPER	animated, wallpaper, film	calculator, call, GPS
WRITE_EXTERNAL_STORAGE	file, video, reader	task, widget
GET_ACCOUNTS	contact, registry, expense	audio, word, sports
GET_TASKS	lock, security, ringtone	calculator, word
KILL_BACKGROUND_PROCESSES	kill, running, task	dictionary, word, weather
WRITE_SETTINGS	alarm, lock, ringtone	book, calculator
CALL_PHONE	call, contact, group	file, joke, game
SEND_SMS	SMS, message	dictionary, sound
WAKE_LOCK	chat, player, radio	bible, dictionary, calculator
VIBRATE	alarm, battery, chat	bible, joke, word
RECEIVE_BOOT_COMPLETED	battery, backup, notification	calculator, dictionary

coarse location permission is used in occasions like weather report and restaurant recommendation. So the influential words in this model are corresponding with practice. Besides this example, words such as "picture" and "photo" are positive to the CAMERA permission, which also fit in with our expectation. So the extraction of influential words from models is also basically correct from empirical analysis. From this result, we also conclude the description of an application has a strong relation with the permission list of this application. Therefore, it is a good way to make a model predicting an app's different permissions based on its description.

In addition, some words occur several times no matter as a positive word or a negative word, such as dictionary, calculator, message and location. This indicates that if this kind of words occurs in the description of an application, the model can predict the permissions correctly to a great extent. Here, we take the word "dictionary" as an instance. The word "dictionary" is positive to ACCESS_WIFI_STATE permission, but is negative to permissions about location and permissions about call and SMS. So if there is a word "dictionary" in an application, we will have a quite high probability to make a correct prediction on different permissions of this application. Therefore, the occurrence of these words greatly contributes to the quality of models. As for these influential words, most of these words, such as map, wallpaper, camera and keyboard, are directly represent one of an application's functions. It can be said the word "map" indicates that one of the application's functions is map. The similar case goes with word "wallpaper" and "keyboard". This also demonstrates the functions of an application closely relate to the actions.

6 Permission Comparator

6.1 Method to Detect Reliability of Permissions

For an application in Android Market, we define the reliability of its permissions as whether this application should request this permission from the perspective of its functions. If an app really needs a permission to accomplish its actual function, we think that the request of this permission is reliable. On the other hand, if an app requests a permission which has no relation with its description, we guess this permission is unreliable.

We can use the model mentioned in Section 3 to detect the reliability of an application's permissions. In detail, if the model which makes a prediction on the real permission list of an application predicts one of the permissions requested by an application should not occur, we believe this permission of this application is not reliable. To apply our model better in detecting the reliability of an application, we should choose a fine threshold of our model to get a higher True Positive Rate without influencing False Positive Rate too much when predicting the reliability of permissions. So from the ROC curves, we find that we can tune the thresholds of different permissions to fulfill the requirement of greater than 90% True Positive Rate and less than 30% False Positive Rate. That is to say, we can use some suitable thresholds to automatically detect the reliability of the permissions of an application. Therefore, our security framework is effective in finding the application with wrongly requested permission list in install-time permission system.

6.2 Test on Real Malware

After choosing a fine threshold of our model, we use this model to predict the reliability of malicious apps' permissions. Here we test this method on a real malware announced by Google.

The malware is Steamy Window, which was announced to carry Android.Pjapps code in February, 2011. A report on this malicious application by Symantec shows that this malware adds several bookmarks to the browser and sends users personal information to some certain server. Besides, it can also send some text messages and block some messages with the number of service provider [14].

In Android Market, there was a legitimate application whose name, description and screenshot were all similar with this malware but behaviors were different. The permission list of the legitimate version application is `INTERNET` and `RECORD_AUDIO`. However, the malicious application's permission list is `INTERNET`, `RECORD_AUDIO`, `RECEIVE_SMS` and `READ_HISTORY_BOOKMARKS`.

After testing this application with our model, the result is that the malicious application should not request `RECEIVE_SMS` permission (we didn't have a test on the permission `READ_HISTORY_BOOKMARKS` for the reason discussed in section 4.1). This result corresponds with the analysis report by Symantec. The `RECEIVE_SMS` permission in the malicious app is used to drop some messages without users' attention. So in this example, `RECEIVE_SMS` permission should not be added in manifest file and is disobey with the ordinary description of this application.

Besides this application, among all the applications announced malicious by Google, many of these malware conceal themselves as an existed trusted application in Android Market. However, these malware change the permission list and add some malicious functions to the ordinary applications.

After testing this malicious application with our model, we think the method predicting the reliability of permissions can also be used in mitigating the security problem of apps in some extent, especially the apps that conceal themselves as some trusted apps. For these malicious applications, some of their permissions are usually additionally added to realize the malicious activity. So we can analyze the reliability of their permission lists based on the relation between the description and the permission list.

7 Conclusion

In this paper, we provide a permission-based abnormal application detection framework which identify an abnormal Android app based on its description and its permission list. This novel framework consists two parts: the model which reflects the relation between the description and the permission list and the permission comparator. In detail, we use machine learning method (Naive Bayes with Multinomial Event Model) to predict the occurrence of different permissions of an application based on its description.

We evaluate our model with 5,685 applications collected from 23 different categories in official application store. The result shows that our model is able to have an accurate prediction on different permissions. Besides, we extract some words that have great influence on different permissions. Furthermore, we define the permission comparator to detect the reliability of the permission list of an application and view the permission list's reliability as the criterion of detecting application with hidden danger. After using this model to test a real malware, we find this method is effective in mitigating the security problem of Android applications, especially the malware that conceals themselves as a legitimate app.

Acknowledgment. This work is partially supported by the HGJ National Significant Science and Technology Projects under Grant No. 2012ZX01039-004-009, Key Lab of Information Network Security, Ministry of Public Security under Grant No.C11606, the National Natural Science Foundation of China under Grant No. 61170263.

References

1. G. Inc., `https://play.google.com/store`
2. (June 5, 2012), `http://www.appbrain.com/stats/number-of-android-apps/`
3. Enck, W., Gilbert, P., Chun, B., Cox, L., Jung, J., McDaniel, P., Sheth, A.: Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, pp. 1–6. USENIX Association (2010)
4. Enck, W., Octeau, D., McDaniel, P., Chaudhuri, S.: A study of android application security. In: Proceedings of the 20th USENIX Security Symposium (August 2011)
5. Enck, W., Ongtang, M., McDaniel, P.: On lightweight mobile phone application certification. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 235–245. ACM (2009)
6. Portokalidis, G., Homburg, P., Anagnostakis, K., Bos, H.: Paranoid android: versatile protection for smartphones. In: Proc. 26th Annual Computer Security Applications Conference (2010)
7. Zhou, Y., Wang, Z., Zhou, W., Jiang, X.: Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In: Proceedings of the 19th Annual Network and Distributed System Security Symposium (2012)
8. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 15–26. ACM (2011)
9. Lewis, D., Gale, W.: A sequential algorithm for training text classifiers. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 3–12. Springer-Verlag New York, Inc. (1994)
10. McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classification. In: AAAI 1998 Workshop on Learning for Text Categorization, vol. 752, pp. 41–48 (1998)
11. G. Inc., `http://developer.android.com/reference/android/Manifest.permission.html`
12. Barrera, D., Kayacik, H., van Oorschot, P., 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, pp. 73–84. ACM (2010)
13. Bradley, A.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145–1159 (1997)
14. Symantec (Februbary 28, 2011), `http://www.symantec.com/connect/blogs/android-threats-getting-steamy`