Identifying malicious Android apps using permissions and system events

Hongmu Han, Ruixuan Li* and Xiwu Gu

School of Computer Science and Technology,
Huazhong University of Science and Technology,
Wuhan, 430074, China
Email: Hanhongmu@mail.hust.edu.cn
Email: rxli@mail.hust.edu.cn
Email: Guxiwu@mail.hust.edu.cn
*Corresponding author

Abstract: With the popularity of the Android platform, more and more hackers take the Android platform as the profitable target. Android provides a risk communication defence mechanism against malicious applications, which has been demonstrated to be ineffective. It is common to quickly identify malicious applications by permission-based analysis methods. Recently, those permission-based methods are becoming useless when more and more applications request dangerous permissions. The proposed approaches are based on the key insight that the difference in the components trigger model in malware applications and benign applications. The malwares are interested in monitoring system broadcast to activate malicious components and request more permissions. The benign applications are preferable to receive self-define broadcast to activate their components and ask fewer permissions. Existing permission-based Android malware check methods can identify nearly 81% malware samples, but they also identify many normal applications as malware. In this paper, we extend the permission-based approach and employ machine learning approaches to identify the malicious applications. We use the datasets of the Market 2011, Market 2012, Market 2013 and Malware to evaluate the proposed methods. The experimental results illustrate the effectiveness of our proposal.

Keywords: Android; malware; risk; application; embedded system.

Reference to this paper should be made as follows: Han, H., Li, R. and Gu, X. (2016) 'Identifying malicious Android apps using permissions and system events', *Int. J. Embedded Systems*, Vol. 8, No. 1, pp.46–58.

Biographical notes: Hongmu Han received his BS and MSc in Computer Science from Wuhan Institute of Technology in 2004 and 2007 respectively. He is currently a PhD student at the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His current research interests are mobile security and cloud security.

Ruixuan Li is a Full Professor of School of Computer Science and Technology at Huazhong University of Science and Technology (HUST), Wuhan, China. His research is focused on cloud computing, big data management and analysis, distributed system security, information retrieval, data mining, social network, peer-to-peer computing, data integration, semantic web and ontology.

Xiwu Gu is a Lecturer of School of Computer Science and Technology at Huazhong University of Science and Technology (HUST), Wuhan, China. His research interests include distributed system security, information retrieval and data mining.

This paper is a revised and expanded version of a paper entitled 'Identifying malicious Android apps using permissions and system events' presented at the Chinese Conference on Trusted Computing and Information Secruity (CTCIS'2014), Enshi, China, 13–15 September 2014.

1 Introduction

Smartphones are becoming increasingly popular, which are taking the place of feature phones in recent years. The most dominant smartphone operating systems are Google Android, Apple iOS and Windows Phone platforms. There are many Android app markets on the internet, and the total number of applications available for users is very huge. However, the popularity of smartphones and the vast number of the applications make the Android platform the most attractive target to hackers. Currently, various forms of the malware applications exist on the Android platform. Moreover, many attack techniques, such as app repacking, denial of service (DoS) attacks (Traynor et al., 2009) and

Copyright © 2016 Inderscience Enterprises Ltd.

privilege escalation attacks (Davi et al., 2011), affect the applications and system components on Android at runtime (Rastogi et al., 2013).

Google Play is growing faster than the Apple App store. However, the application security check in Google Play is not as good as the Apple App store, which may check each available application manually by software security experts. Google provided a security feature, called application verification service, to protect against malicious Android applications on November 13, 2012. A study shows that the application verification service fails to stand up its task, which misses 85% malwares that are already known to be malicious (Jiang, 2012). Furthermore, Android allows users to install third-party applications on the smartphones. It increases the risk of infecting malwares.

The Android security model is based on application-oriented mandatory access control and sandboxing (Barrera et al., 2010; Bartel et al., 2012). Each application assigns a unique User ID and a set of permissions at the app installation time, which cannot be changed afterwards before Android 4.3. There are more than 130 permissions that govern accessing system resources or communicating with other applications. The Android permission system checks corresponding permission assignments at runtime. If an application has no appropriate permissions, it will not be allowed to access the privileged resources. Users must approve the application permission request at the app installation stage if they want to run the application on their smartphones. 'App ops' is a hidden feature in Android 4.3, which can control the permissions for the individual app after the app installed. However, without extra information, the users find it hard to determine which permission is harmful since they have rare specific knowledge of Android permissions. With this vulnerability, many applications request more additional permissions than they actually need on the Android platform. Recently, several studies have explored that abusing permission request can lead a serious risk to Android users (Felt et al., 2011a; Peng et al., 2012; Sarma et al., 2012). With the increasing capability of the smartphones, the Android applications are becoming more and more complicated, and require more dangerous permissions than before. We find that only using permissions to reveal the malicious Android applications is insufficient.

There are several analytical techniques to detect malwares on Android (Shimada et al., 2011; Chakradeo et al., 2013). Most analytical techniques can be classified into two categories: static and dynamic methods. Each detection method has its benefits and drawbacks. Various obfuscation techniques can make malware evade static analysis, while dynamic analysis techniques cannot guarantee a complete coverage over all possible conditions to activate app functions of the program. Without a main function, Android applications employ an event trigger mechanism to invoke their components. It provides a base class called intent that is used by event sources to pass event information. Through intent messaging mechanism, Android can easily realise the late runtime binding between components in the same or different applications. As with the computer virus, the Android malware often perform some type of harmful activity on infected mobiles, such as stealing mobile information, sending premium SMS, monitoring telephone conversations, and so on. The Android malware should request correspond permissions to carry out privilege operation and register some broadcast receivers to monitor system behaviour. Some researchers have revealed that many Android malicious programs, such as DroidKunFu (Zhou et al., 2012) and DroidDream (Security 2011.3) infect applications and add receiver components and service components. For example, a malware can start a background service to intercept a short message (SMS) (Zhou and Jiang 2012), it must have a broadcast to monitor SMS and two permissions to receive and send SMS. Through analysing the Malware datasets, we can see that malicious applications are not only requesting more permissions but also monitor more system events than normal applications.

Many permission-based methods have been proposed to identify high risk applications. Enck et al. (2009) propose triggering a risk warning if an application requests certain permission or combination of two or three permissions. Sarma et al. (2012) analyse the risk of Android applications according to the permissions that an app requests, the category of the app, and what permissions are requested by other applications in the same category. If an app requests a critical permission that is rarely requested by other applications, it will be deemed to be a risky app. In Peng et al. (2012), risk scoring and risk ranking are proposed to improve risk communication for Android applications. Mobile payment is a critical problem in mobile, a secure mobile payments framework has been proposed based on universal integrated circuit card (Ahamad et al., 2014). Magat et al. (1988) show that frequent warnings desensitise users, especially if most warnings do not lead to negative consequences. Some users are not likely to pay attention to install-time permission warning in the Android platform. In addition, the current Android warning approach describing the risk information of each app is a 'stand-alone' function that requires too much technical knowledge and time to retrieve useful information. In this paper, we employ K-means clustering and outlier detection algorithm to analyse the risks of Android application permissions. The experimental results show that the Android permission analysis method is very suitable to the dataset of the Market 2011 and Market 2012. However, it is under a high false positive rate (FPR) in Market 2013. We carefully analyse Market 2013, and find that the applications in Market 2013 are becoming more complicated than the previous two market datasets. This causes applications in Market 2013 to request more and more permissions that make permission-based analytical approaches become ineffective.

To improve the capability of identifying malicious applications, we pay attention to the Android system event messages. More than 86% malware samples are repackaged in malware dataset (Zhou and Jiang, 2012). Most of those repackaged legitimate apps include malicious payloads by adding broadcast receiver components and service components. Broadcast receiver components are used to monitor system events and execute preliminary work, such as launch malicious service in background, or intercept SMS spread to other components. The service components response to perform some malicious operations in background, i.e., sending premium SMS, and recording users' phone call. Plenty of non-repackaged malwares are also likely to take the similar method to do malicious operation. It is necessary to take broadcast receiver as feature vector to distinguish malwares from normal Android applications. To verify our suspicion, we take libsvm (Chang and Lin, 2011) as a tool to distinguish the malware applications from normal applications by analysing the action list of all intent filters of application broadcast receivers. The experimental results demonstrate that our suspicion is correct. In summary, the contributions of this paper are as follows:

- We analyse the existing permission-based risk analysis methods, and figure out their merits and faults.
- We analyse the existing permission-based risk analysis methods, and figure out their merits and faults.
- We extend the existing permission-based risk analysis approach and add intent as a feature vector. We employ the support vector machine (SVM) approach, and propose intent-based detection (IBD) and intent-permission-based detection (IPBD) methods to identify the malicious applications. The experimental results show that our methods not only have a highly precise ratio to identify malware, but also have a low false positive ratio during analysing benign applications.

The rest of this paper organised as follows. We present a description of the threat model for the Android platform and the current risk warning mechanism in Section 2. Section 3 discusses the datasets that we have collected. In Section 4, we discuss the merits and faults of existing permission-based Android malicious analysis methods. We propose two methods that take intent and intent-permission as feature vectors respectively to distinguish malwares from benign applications. We then present experimental results in Section 5. Finally, we discuss the related work in Section 6, and conclude the paper in Section 7.

2 Android overview

To defend against the malwares in the Android platform ecosystem, the Android system provides two major methods: sandboxing each application and warning users about the permissions that the application is requesting (Blasing et al., 2010). However, many studies show that those defence mechanisms are insufficient to prevent a user from using malicious applications on the Android platform. In this paper, we focus on the malicious Android applications with a sophisticated application broadcast mechanism and coarse-grained permission model.

2.1 Adversary model

As the most popular smartphone platform, there is a wide array of application markets on Android that contains a large number of applications, and a user may install applications from varying origins. Users install applications with unknown developers or distrust origins may lead to serious security threat. The malwares are likely to pretend hot applications on Android. Repackaging is a common technique that is widely used by hackers on the Android platform. Usually, the hacker downloads a hot application, decodes the application, modifies permissions declaration, adds receiver components and service components. Broadcast is part of the Android platform communication mechanism. With broadcast receivers, malicious applications can monitor the application operations and launch a self-defined component. Then he repackages this modified application, and distributes it on the app markets. The malwares can utilise the dangerous permissions to carry out the malicious behaviours (Felt et al., 2011a; Wei et al., 2012). But recently, the Android applications are becoming more and more complicated than before. They need to request much more dangerous permissions and make permission-based analysis methods become useless. As existing malwares on the PC, the malicious applications also prefer to implement malicious behaviours on background. Because of the event trigger mechanism on Android, the malwares must get some events to activate their malicious components. The best way to accomplish this target is to use the broadcast receiver to trigger malicious component in the background. According to our observer, most malwares request many dangerous permissions and use broadcast receivers to launch malicious components, small part of malwares embed malicious operations in activities. In this paper, we focus to identify the former, the later need to analyse their codes. We expect to find a method which has high precise ratio and low error ratio to distinguish different applications on Android.

2.2 Android permission model

The Android security model differs significantly from the PC security model. Android applications treat each other with distrust principle. The applications are isolated from each other and do not have access to the private data of other applications. With suitable permissions, Android applications can access personal and sensitive information such as contact lists, phone numbers, geographic location, and SMS messages on the smartphone. To protect the Android system resources from illegal access, Android platform classifies the permissions into four protection levels.

- 'Normal' permissions protect access to API calls that are harmless to the users. Those permissions could be granted automatically, without being prompted to the users.
- 'Dangerous' permissions control access to dangerous API calls. These permissions will be presented to the

users when the application is installed and the users will be asked to consent to grant them.

- ‘Signature’ permissions represent the highest privilege. These permissions can be obtained only if the request application has the device manufacturer’s certificate.
- ‘SignatureOrSystem’ permissions are specially signed permissions, and also have the highest privilege. Only the applications that are in the Android system image or signed with the same certificate in the system image could be granted with the SignatureOrSystem permissions.

According to the protection level definition, third-party applications can only use normal and dangerous permissions. Only the manufacturer or system preinstalled applications can use the four protection level permissions.

2.3 Android message mechanism

Growth of smartphone scale has been a prevalent trend of the modern mobile market in the world. The user could get Android applications from various origins, such as Google Play, third-party app markets, forums, SNS and so on. This makes it hard to protect smartphones from malware applications. Under the Android security model, it provides a sandbox mechanism to protect the applications. Each application is allocated a unique user ID and a group ID. The application can only access its own files by default. Sensitive information of the Android applications can be protected from other applications by the isolation mechanism. To implement inter-application communication, Android provides a message passing mechanism where intents are used to link applications. The intent is an object that contains the component name, action, category, data, and other attribution. It can be utilised to both inter-application communication and intra-application communication. According to the difference of the recipient, intent can be classified into two types: explicit intent and implicit intent. An explicit intent defines component name attribute to identify the intended recipient, while an implicit intent leaves it up to the system to determine which applications should receive the intent.

Activities, services, broadcast receivers and content providers are Android application components, which are logical application building blocks. The explicit intent and implicit intent could be exploited to start activities; start, stop, and bind services; and broadcast information to broadcast receivers. If a component is accidently declared public and incorrectly defines its intent filter, the external applications can invoke its component in unusual ways or inject malicious data into it by faking external intent (Chin et al., 2011). There is no main function in Android application, instead a main activity as an entry point. The Android applications employ event triggers, and different actions could trigger different events. A touch event may cause a components callback function, while a system event will trigger a system broadcast message. For example, an event of boot-completed will send a broadcast intent. For the Android event trigger mechanism, monitoring system event messages can reveal various system actions and launch different components. Many malwares are likely to receive a broadcast of boot-completed and launch a background service without notifying the user. To receive a system broadcast, the application should register one or more broadcast receivers. A broadcast receiver’s declaration decides what broadcast intent will be sent to the broadcast receiver. Repackaging a hot application is easier than spreading the malicious applications. The hacker can perform his purpose by adding one or more components, such as broadcast receivers, activities and services. In this paper, we carefully analyse the broadcast receiver’s declaration on different datasets, and try to find what system event messages are malware interested in the Android platform.

3 Datasets

In this section, we describe the four datasets that we used in our study of Android application permissions, and we use two of the datasets to analyse the application event broadcast receivers.

3.1 Dataset description

3.1.1 The datasets of Android permissions

- *Market datasets*. We obtained the permission dataset of Market 2011 and Market 2012 from the authors (Peng et al., 2012). The first dataset, Market 2011, consists of 157,856 app permissions in February 2011. The second dataset, Market 2012, consists of 324,658 app permissions in February 2012. Both datasets are obtained from the Google Play. In addition, we download 3,706 top free applications from Google Play in September 2013 randomly. We called this app dataset Market 2013. We use four types of antivirus tools, 360 internet security, Kaspersky, AVAST and Norton, to analyse Market 2013, and get rid of warning applications from Market 2013.
- *Malware dataset.* We obtained the Malware dataset from the authors of Zhou and Jiang (2012), which consists of 1,482 malware applications. Next, we remove the duplicate applications in the Malware dataset and extract the permission descriptions from each app. Finally, we get 209 unique malware app permissions.

3.1.2 The datasets of broadcast receivers

- *Market 2013.* We collected the action list from application broadcast receivers in Market 2013. The action string indicates what intent the broadcast can receive.
- *Malware dataset.* We extract the action list from application broadcast receivers in Malware dataset. It was mentioned above.

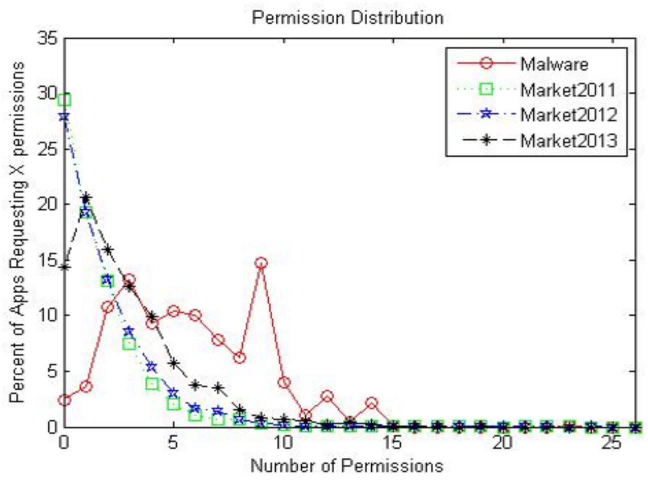
Market 2011 and Market 2012 consist of application permission data. Therefore, we do not extract the action list from these datasets.

3.2 *Data fetching and feature selection*

Market 2011 and Market 2012 are permission datasets that have been cleaned by the authors, Peng et al. (2012). We extracted the file Menifest.xml from each app in Market 2013 and Malware dataset. Through parsing the manifest file, we got a permission list and an action list for each app.

We take 26 permissions as feature vectors for our research, the same with (Peng et al., 2012; Sarma et al., 2012). Figure 1 shows the percentage of applications that request different numbers of the 26 permissions. From the figure, we notice that malicious applications tend to request more permissions than those in the market datasets. From the Market 2011, Market 2012 and Market 2013, we find a tendency that applications are requesting a number of permissions on average are increasing. Figure 1 shows that the applications in Market 2013 request more permissions than before, which make it become hardly to distinguish normal applications from malwares. In general, the analysis results demonstrate the close link between malware and permissions in Android platform where the malicious applications are requesting permissions in quite different ways compared with normal applications.

Figure 1 The percentage of applications that request a specific number of 26 permissions for different datasets (see online version for colours)



Activities, services and broadcast receivers are the three main components on Android. Each of them can be taken as an entry point to run an app. If the attributes of an intent are consistent with a broadcast receiver declaration, it can activate the broadcast receiver. The action and data are the primary attributes of the intent. In this paper, we focus on the action in broadcast receiver, where the Android system provides more than 100 category actions. In addition, the developers can add new actions for their applications. Since each component is an entry point, broadcast receivers can be exploited to launch a malicious component. For example, a malicious app can launch a service component in the background when it receives a boot-completed broadcast to record the user operation. To receive the broadcast, an application should statically register broadcast receivers through the ¡receiver¿ tag in AndroidManifest.xml, or dynamically registers broadcast receivers by using the function of registerReceiver () in the programming code. The latter has a fault that a broadcast receiver could obtain the event message only when the app is running.

Table 1 The relationship of applications, broadcast receivers and category actions in Market 2013 and Malware datasets

Datasets	*Number of applications*	*Number of broadcast receivers*	*Number of category actions*
Market 2013	3,706	5,818	2,559
Malware	1,482	1,786	117

Table 2 The percentage of action usage in the broadcast receiver of Market 2013 and Malware datasets

Actions	*Market2013*	*Malware*
BOOT_COMPLETED	23.69%	89.60%
SIG_STR	0.00%	40.00%
BATTERY_CHANGED_ACTION	0.00%	35.48%
SMS_RECEIVED	2.02%	33.25%
CONNECTIVITY_CHANGE	5.59%	24.05%
USER_PRESENT	3.40%	21.98%
ACTION_POWER_CONNECTED	1.40%	21.98%
INPUT_METHOD_CHANGED	0.00%	21.98%
PICK_WIFI_WORK	0.00%	14.84%
UMS_CONNECTED	0.08%	14.84%
UMS_DISCONNECTED	0.03%	14.84%
MEDIA_NOFS	0.00%	14.76%
PHONE_STATE	2.94%	9.44%
WAP_PUSH_RECEIVED	0.27%	5.63%
NEW_OUTGOING_CALL	2.05%	4.52%
APPWIDGET_UPDATE	33.78%	4.21%
PACKAGE_ADDED	10.04%	2.86%
PACKAGE_REMOVED	8.12%	2.62%
INSTALL_REFERRER	15.00%	2.46%
PACKAGE_REPLACED	9.12%	2.46%
MEDIA_MOUNTED	1.27%	0.08%

With the rapid spread of repacking technique on Android, more and more hackers choose to decode hot applications, add service and broadcast receivers, and modify the manifest file to produce malware. Therefore, we extract the declaration of broadcast receiver in manifest file as our original analysis data. Table 1 shows the relationship of applications, broadcast receivers and category actions in Market 2013 and Malware datasets. From Table 1, we observe that malicious applications are more interested in receiving the system event broadcast than the ones in the market datasets, while the normal applications are more likely to receive self-defined broadcasts. Different system event broadcast indicates different event happened. Android takes intents to transfer event messages. For example, an SMS_RECEIVED coupled with a sticky broadcast intent indicates that an SMS has been received. The SMS_RECEIVED used by 33.25% of the malicious applications, but only 2.02% applications of the Market 2013. It was suggested that the reason might be the malware applications attempt to block the sticky broadcast and intercept messages of the user applications. None of broadcast receivers obtain the intent with the action of INPUT_METHOD_CHANGED in Market 2013, but 21.98% applications in Malware do. The reason is that some malware tries to record users inputs, such as accounts, passwords and other private information.

Table 2 shows the top 21 most used actions in the broadcast receivers in the two datasets Market 2013 and Malware. Table 2, column 2 shows the percentage of actions in the broadcast receiver of the Market 2013, and column 3 shows the percentage of the Malware dataset. There are 21 actions in Table 2, but we choose 16 actions as the feature vectors to identify the different applications. The reason is that some actions do not exist in the Market 2013 dataset. We believe that using these actions will influence the results. To avoid over-fitting our model, we do not take those actions as the feature vectors to identify malware applications.

3.3 Data discussion

On Android platform, each application runs with a distinct system identifier, i.e., user ID and group ID. It is isolated from other applications and the system. The feature of privilege-separated operating system is the characteristic of Android. In addition, Android provides a fine-grained security feature that uses a permission mechanism to enforce restrictions on the sensitive system resources. Intents are used to activate components, send messages, and transfer data. In order to receive some special intent, the components not only need to declare suitable intent filters, but also must get some specific permission. For example, an application can receive a broadcast intent with BOOT_COMPLETED only if it has a permission of RECEIVE_BOOT_COMPLETED. Those actions must accompany the corresponding permission. According to our observation, the distribution of those permissions and actions are quite consistent in different datasets. Therefore, we are confident to believe that the purpose of those applications requesting the permissions and declaring the same actions for broadcast receiver is that when a malicious application received system event message, it can start corresponding operations.

4 Detection methods

4.1 Permission-based K-means clustering and outlier detection

Since the permissions are very important to analyse the malware applications on Android, we take the permission-based analysis method as the baseline to identify the malicious applications. We analyse Market 2011, Market 2012 and Malware datasets, and find two phenomena. First, the datasets are unbalanced since the size of Market 2011 and Market 2012 is several orders of magnitude larger than that of the Malware dataset. Second, malicious applications request more permissions than the benign applications. Because of the above two features, we employ K-means clustering and outlier detection to differentiate malwares from benign applications.

4.1.1 K-means clustering

We merge Market 2011 and Malware dataset into one dataset, and randomly select part of this dataset as a training dataset. The dataset is represented as (X_i, Y_i), $i = 1, 2, \cdots, n$, where X_i represents a n-dimensional vector $(x_1, x_2, \cdots, x_n)$ and $Y_i = -1, 1$ represents the corresponding class label with 1 for benign and –1 for malware. For K-means clustering, we set the input parameter k as the number of clusters, and partition the training dataset that contains n application permissions into k clusters (Hartigan and Wong, 1979).The k clusters have two characteristics: the intra cluster similarity is high, but the inter cluster similarity is low. The mean value of the object similarity in a cluster is defined as the cluster similarity, which is the cluster centroid or the centre of gravity. We use the weighted Euclidean distance to donate the similarity between two applications. It is computed as follows:

$$d(i, j) = \sqrt{\omega_1 \left|x_{i1} - x_{j1}\right|^2 + \cdots + \omega_n \left|x_{in} - x_{jn}\right|^2} \tag{1}$$

where the permissions of app_i are denoted as $(x_{i1}, x_{i2}, \cdots, x_{in})$. If app_i has the permission j, then x_{i1} equals 1, otherwise the x_{i1} equals 0. We take the square-error criterion to compute the criterion function, which is defined as follows:

$$S = \sum_{i=1}^{m} \sum_{o \in C_i} \left|o - avg_i\right|^2 \tag{2}$$

where S is the sum of the square-error for all objects in the dataset; o is the point in the space representing a given object; and avg_i is the mean of the cluster C_i (Han and Kamber, 2006). To save the space, we do not list the detailed operational flow of the K-means algorithm.

4.1.2 Outlier detection

We defined outlier as the malicious app that is inconsistent with the most application data. The outliers can be mainly classified into two categories:

- External outlier: The outlier is an element of the application cluster that has few applications, and is located far from other application clusters.
- Internal outlier: If an application locates far from the centre of the cluster by comparing the distance of the other data in the same cluster, the application is an outlier.

According the concept of outlier, we take outlier factor to describe the difference between the outlier and the normal application. The outlier factor is defined as follows:

Definition 4.1: If the dataset D is partitioned into k clusters C_1, C_2, ⋯, C_k by K-means clustering algorithm, the outlier factor of object p is defined as the average of the weighted distances between p and cluster centres, denoted as $OF(p)$.

$$OF(p) = \sum_{i=1}^{k} \frac{|C_i|}{|D|} \cdot d\left(p, C_i\right) \tag{3}$$

where $d(p, C_i)$ is the distance between application p and the centre of cluster i.

To identify the malwares, we make the simplifying assumption that outlier factors are independent and follow a normal distribution. Then we can easily identify outliers using the following definition.

Definition 4.2: $\vee p \in D$, where p is an application, and D is an Android application permissions dataset, compute the outlier factor $OF(p)$ for object p, the average of outlier factor Avg_OF and the standard deviation of outlier factor Dev_OF. If the application is not a malware, the $OF(p)$ must subject to the threshold value that is defined as follows:

$$OF(P) < Avg\_OF + \beta \cdot Dev\_OF (1 < \beta < 3) \tag{4}$$

where β is a constant. Usually, $\beta = 1$ or 1.285 (Lowe, 1999). For normal distribution, there is a 68-95-99.7 rule, or the empirical rule. According to the permissions distribution in Figure 1, the value of β to 1.285 is better. Permissions data distribution is approximately normal, then about 80% of the data values are within 1.285 standard deviations of the mean. The K-means clustering and outlier detection algorithm can be summarised as follows (see Algorithm 1).

4.2 IBD and IPBD

In Android system, three components, activities, services and broadcast receivers, can be the entry points to analyse the Android applications. The intent is the message to activate these components in an asynchronous way. The action string is the primary attribute to an intent object. It is composed of a verb that indicates the action to be performed. To broadcast intent, the action indicates what has happened and what is reported. We use action list to distinguish different intents that can be received by broadcast receivers. We implement our methods on Market 2013 and Malware datasets. To quickly identify the malwares, we employ a SVM as a tool to analyse the datasets instead of using the outlier detection method since the two datasets are very close.

Algorithm 1 k-means clustering and outlier detection algorithm

```
Input:
Market 2011, Market 2012, Market 2013 and Malware datasets.
Output:
Detection rate and warning rate.
Method:
1:  for $k = 1; k \le m; k++$ do
2:      $TrainData = DividData(Market2011, Maearedatasets)$;
3:      $C = K - means(TrainData, k)$;
4:      for $j = 1; k \le n; j++$ do
5:          $OF(p_j) = \sum_{i=1}^{k} \frac{|C_i|}{|D|} \cdot d(p_j, C_i)$
            $p_j \in TrainData$
6:      end for
7:      $Avg\_OF = \frac{\sum_{j=1}^{n} OF(p_j)}{n}$
8:      $Dev\_OF = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (OF(p_i) - Abg\_OF)^2}$
9:      $Threshold = Avg\_OF + \beta \cdot Dev\_OF \ (1 \le \beta \le 2)$
10:     $WarnRate2011[k] = Detection(Market2011, C, Threshold)$;
11:     $WarnRate2012[k] = Detection(Market2012, C, Threshold)$;
12:     $WarnRate2013[k] = Detection(Market2013, C, Threshold)$;
13:     $DetectionRate[k] = Detection(Malware, C, Threshold);)$
14: end for
```

4.2.1 Support vector machines

SVM (Han and Kamber, 2006) is a popular machine learning algorithm that has been widely applied in classification, regression and novelty detection. SVM approaches use a hyperplane to classify the data into different classes. It needs to search for a linear optimal separation hyperplane. This problem is that the determination of the model parameters corresponds to a convex optimisation problem by searching for the maximum marginal hyperplane. For a training set of instance-label pairs (x_i, y_i), $i = 1, \cdots, m$, where $x_i \in R_n$, and $y \in \{-1, 1\}^m$, the SVM can be expressed as an optimisation problem:

$$\min_{\omega,b,\xi} \frac{1}{2} \omega^T \omega + C \sum_{i=1}^{m} \xi_i \qquad C > 0 \tag{5}$$

Subject to

$$y_i \left(\omega^T f\left(x_i\right) + b\right) \ge 1 - \xi \tag{6}$$

where $\xi_i \geq 0$; the constant C is the penalty parameter, which belongs to the error term; the function f maps training vector x_i into a higher dimensional space. Commonly, the kernel function is defined as follows.

$$K(x_i, x_j) \equiv f(x_i)^T f(x_j) \tag{7}$$

There are four common kernel functions:

- Linear: $K(x_i, x_j) \equiv x_i^T x_j$
- Polynomial: $K(x_i, x_j) \equiv \left(\lambda x_i^T x_j + \alpha\right)^d, \alpha > 0$
- Radial basis function (RBF): $K(x_i, x_j) \equiv \exp(-\alpha \| x_i - x_j \|^2), \alpha > 0$
- Sigmoid: $K(x_i, x_j) \equiv \tanh\left(\alpha x_i^T x_j + \gamma\right), \alpha > 0$

where α, γ and d are kernel parameters. In our experiments, we do not get the manifest sets of Market 2011 and Market 2012. Therefore, we take Market 2013 and Malware datasets to evaluate our methods.

The SVM algorithm depends on the selected feature vectors. From Section 3, we can find that the app in different datasets has a quite different interest. The malicious applications request more permissions and receive more system event messages than normal applications. In addition, the selected feature vectors should avoid over-fitting prediction model. We take broadcast intent and intent-permission as feature vectors respectively, and propose IBD and IPBD methods to identify the malicious applications. However, there are many attributes in an intent object. This will make our feature vectors become obscure. Consequently, we only take the action attribute as the feature vector for intent since it is the primary attribute of intent and has special semantics. We extend LibSVM (Chang and Lin, 2011), an SVM package, to implement our methods by utilising the SVM variation and an RBF kernel. For SVM, it is difficult to find the best prediction model. We used n-fold cross-validation to locate the optimum SVM parameters. With those parameters we can get the optimum prediction model.

The algorithms for IBD and IPBD can be described as follows.

Step 1 We use the development tools to extract and parse the manifest files from the app files, i.e., the apk files. After that, we get an action list and a permission list from each app.

Step 2 The action dataset (or action-permission dataset) is used to train a multivariate classifier. Our SVM takes a RBF as kernel function.

Step 3 We use LibSVM to implement the data analysis function. The software executes the following step.

1. Train SVM using a subset of dataset (action dataset or action-permission dataset).
2. Compute cross-validation accuracy for Step 1.
3. Get the best parameters for LibSVM from the best cross-validation accuracy.
4. Compute a prediction model.
5. Predict datasets using the prediction model.

Step 4 Compute the true positive rate (TPR) and FPR for IBD and IPBD. Calculate detection rate and warning rate for Market 2013 and Malware datasets.

5 Evaluation and discussion

In this section, we performed several experiments to illustrate the effectiveness of our malware detection methods. In Section 5.1, we present the schemes of three experiments. We describe and analyse the results of the experiments in Section 5.2.

5.1 Agenda of experiments

The following set of experiments was performed to evaluate the performance of the malware detection approaches on different feature selections. We prepared three kinds of datasets for the three experiments respectively. The first kind of dataset consists of 26 permissions. The second category dataset contains 16 actions. The last one is composed of 26 permissions and 16 actions.

We assume that all applications in the Market 2013 dataset are benign though a few of them may be malicious applications. We randomly select 40% applications of Market 2013 and Malware datasets to consist of the training set, and the rest is the testing set.

Experiment 1: The purpose of Experiment 1 is to evaluate the permission-based k-means clustering and outlier detection. The experiment seeks to exploit the difference in Android permissions to distinguish malwares.

For the training set, we first compute the weights of permissions, which will be utilised to compute the dissimilarity matrix in K-means clustering algorithm. Then, we employ K-means and outlier detection algorithm to obtain the cluster centres, outlier factors and outlier threshold values. Finally, we can pick up the results of the previous step as a criterion to predict other applications. For K-means clustering algorithm, we iterate 20 times to find the best cluster number. The partial results can show in Table 3.

Experiment 2: The goal of Experiment 2 is to evaluate the ability of IBD method to identify malwares. We leverage a SVM tool, named LibSVM, to implement our experiment. To obtain the best prediction model, we take a ten-fold cross-validation to find the optimum SVM parameters in the experiment.

Experiment 3: The purpose of Experiment 3 is to evaluate the ability of the IPBD method to see whether it can overcome the drawbacks of the permission-based detection (PBD) method. The experimental process is the same as to

Experiment 2. However, the feature vectors include not only permissions, but also broadcast receivers.

The experimental results of Experiments 2 and 3 can be seen in Table 4 and Table 5.

5.2 Experimental results analysis

5.2.1 Permission-based malware analysis

An app can execute privileged operations and access sensitive resources only if the corresponding permissions were approved when the app was installed. However, many applications request more permissions than they actually need. One reason is that an under-privileged app may raise security exceptions, whereas an app with over-privileges has no side effect from the apps point of view. In addition, the application developers without sufficient documents cannot request suitable permissions. According to our analysis, we find that the malicious applications request 12 permissions on average in Malware dataset, and the normal applications in the three Market datasets request only two permissions on average. Our analysis on 26 key dangerous permissions is very close to Peng et al. (2012) and Sarma et al. (2012). These permissions are requested by malicious and benign applications. According to our survey on Android app permissions and the datasets, we choose K-means clustering and outlier detection to identify the applications that have a high security risk.

Table 3 presents the results of using K-means clustering and outlier detection to identify malicious applications. Column 1 sets the number of clusters. The k-means cluster is necessary for users to specify *k*, the number of clusters, in advance can be viewed as a disadvantage. It is difficult to decide which K is best parameter before execute the K-means cluster. To find an optimum parameter of K-means cluster, we set the number of clusters from 1 to 20. Column 2 is the outlier detection threshold that was discussed in Section 4. As showed in Table 3, the experiments show that the parameter of K-means cluster has a little impact on outlier detection. Although the market dataset consists of different types of application, they have a few of difference in requesting permissions. The number of malware dataset is far less than market dataset. So the cluster number is not critical in the experiment. However, the experimental results depend on the threshold value. So the threshold value might have to strike a balance between detection ratio and warning ratio. From Figure 1, we can easily find that nearly 20% in malware dataset do not request more permissions than those market datasets. We make the simplifying assumption that outlier factors are independent and follow a normal distribution. According to the empirical rule in normal distribution, we make the value of to 1.285. Thus, the detection ratio of 81.34% is reasonable. When rapidly increasing the number of permissions on average from Market 2011 to Market 2013 in Figure 1, we can find that the warning ratio also increases sharply from Market 2011 to Market 2013 in Table 3. There are a number of multi-function applications emerging on Android application markets. These applications require numerous permissions to implement their multi-functions. This leads to false positive ratio increasing sharply on those permissions-based analytical approaches. They are useless to analyse the recent multi-permission Android applications.

5.2.2 Intent-based and intent-permission based malware analysis

An intent message must be sent to the component to activate a component. The message may come from another component or a system event broadcast. If an application has a receiver to monitor the system broadcast, it can launch one or more components without the users input. Some studies show that malwares wish to use the above method to launch components. In this paper, we try to use the broadcast intent to identify the malicious applications. We take the top 16 actions within intent filters of broadcast receivers to distinguish the applications in different datasets.

Table 3 The percentages of applications identified as malicious by permission-based K-means clustering and outlier detection

Number of clusters	*Threshold*	*Malware*	*Market2011*	*Market2012*	*Market2013*
$K = 1$	$\beta=1$	85.65%	10.40%	12.88%	24.28%
	$\beta=1.285$	81.34%	8.18%	10.45%	18.94%
$K = 3$	$\beta=1$	85.65%	10.44%	12.92%	24.28%
	$\beta=1.285$	81.34%	8.16%	10.44%	19.94%
$K = 5$	$\beta=1$	85.65%	10.40%	12.88%	24.28%
	$\beta=1.285$	81.34%	8.15%	10.43%	18.94%
$K = 7$	$\beta=1$	85.65%	10.41%	12.88%	24.28%
	$\beta=1.285$	81.34%	8.15%	10.43%	18.94%
$K = 15$	$\beta=1$	85.65%	10.46%	12.94%	24.99%
	$\beta=1.285$	81.34%	8.15%	10.43%	18.94%
$K = 20$	$\beta=1$	85.65%	10.43%	12.90%	24.28%
	$\beta=1.285$	81.34%	8.15%	10.43%	18.94%

Table 4 The true positive ratio, false positive ratio in Experiment 2 and Experiment 3

Methods	*True positive ratio*	*False positive ratio*
IBD	85.25%	7.12%
IPBD	93.07%	1.13%

Table 5 The percentages of applications identified as high risk applications

Methods	*Detection rate in Malware*	*Warning rate in Market2013*
IBD	85.24%	7.12%
IPBD	93.07%	1.13%
CRCP	88.42%	(NA)
PBD	81%	(NA)

We take the true positive ratio and false positive ratio to evaluate the prediction model in Experiments 2 and 3. Table 4 shows the experimental results of IBD and IPBD methods. From Table 4, we can clearly find that take intent-permission as feature vectors can achieve a better result.

We compared our IBD and IPBD methods with CRCP (Sarma et al., 2012) and PBD (Huang et al., 2013) through Experiments 2 and 3. Table 5 shows the evaluation results of using IBD and IPBD, CRCP and PBD to discover malicious applications. We define the warning rate is the ratio of the applications identified as malicious by our methods in market dataset. The detection rate is defined as the ratio of the applications detected as malicious by our methods in Malware dataset. In the first row in Table 5, we see that the detection rate in Malware dataset is 85.24%, and the warning rate in Market 2013 is 7.12%. The reason is that the applications in the Market 2013 dataset receive fewer system broadcasts than the applications in Malware dataset. Most of them are more interested in receiving self-defined broadcast. The second row of Table 5 demonstrates the result of the approach based on intent-permission. Note that the IPBD method identifies 93.07% of Malware dataset with low warning rate of 1.13% in Market 2013. The CRCP (Sarma et al. 2012) identified 88.42% of malicious applications (121) and warned 4.9% Market 2011 dataset. From Figure 1, we can see there was a clear tendency that the applications are requesting a greater number of permissions on average in the last three years. Existing approaches consider only the numbers and risks of the permissions requested by an app. Many of permission-based methods are monotonic with the number of permissions. This feature of permission-based methods may define various benign applications as malwares. The IPBD method has a higher detection rate and a lower warning rate than CRCP. Huang et al. (2013) used four machine learning algorithms to evaluate the performance of PBD of Android malwares. They can detect more than 81% of malicious samples.

We use the same Malware dataset that originates from Zhou and Jiang (2012). Another advantage of IPBD is that the false positive ratio of our approach will not increase while increasing in the number of permissions. Therefore, we can conclude that the IPBD method can be used as a quick and stable filter to identify the malicious applications in the applications markets.

6 Related work

6.1 Static and dynamic analysis

With the spread of malware on the Android platform, more and more researchers focus on this field (Felt et al., 2011b; Zhou et al., 2012). Many existing methods are transplanted to analyse the malware on the Android platform. Andromaly (Shabtai et al., 2012) is a host-based malicious applications detection framework that applies machine learning detector to identify malicious applications. However, the authors did not take a real malware dataset to verify their detection framework. DroidAlarm (Zhongyang et al., 2013) is a static analysis tool that can identify potential capability leaks and present concrete capability leak paths in Android applications. Static analysis is a common technique of analysing the Android applications (Schmidt et al., 2009). DroidScope (Yan and Yin 2012) is seamlessly reconstructing the semantics of the system call and Java. However, it transplants the tradition of virtualisation-based malware analysis on Android, without considering the programming model and permission mechanism. TaintDroid (Enck et al., 2010) is a system-wide dynamic taint tracking and analysis tool, which can track multiple sources of sensitive data. RiskRanker (Grace et al., 2012) is a malware detector that assesses untrusted applications revealing potential security risks. Crowdroid (Burguera et al., 2011) proposes a behaviour-based detection framework supporting dynamic analysis on Linux Kernel system calls. However, it only refines the existing syscall-based analysis techniques that are poorly suited for Android. The reason is the fact that it could not capture critical interactions between the applications and the Android system.

To use the static analysis approach, the researchers often suffer from learning program logic. Moreover, there are some techniques against using the static analysis approach, such as Java reflection, encrypt, native develop and randomising profiles (Shastry et al., 2012; Zhongyang et al., 2013). Those techniques cause static analysis to become very hard to use. Dynamic analysis is likewise a very useful analysis technique, whereas there is a gap between the system call and the application behaviour (Schmidt et al., 2008).

6.2 Permission analysis

Felt et al. (2011c) analyse the merits and drawbacks between existing time-of-use and install-time permission system. The latter could have a positive impact on system security when it needs to be declared upfront by the developer. Nevertheless, it can be optimised. Wei et al. (Wei et al., 2012) comprehensively analyse the permission evolution and usage for the entire Android ecosystem. Stowaway (Felt et al., 2011a) is an automated testing tool on the Android API that can detect application over-privilege. However, it cannot deal with Java reflection, which is widely used by crackers on Android development. Self-organising map (SOM) was utilised by Barrera et al. (2010) for empirical analysis of permission-based security model on Android. Jeon et al. (2012) introduce a framework that applies fine-grained access control for Android applications by adding finer-grained permissions. A location-based with time-constraint RBAC (MLT-RBAC) is proposed to access the wireless associated database system for mobile applications (Chen et al., 2012). Apex (Nauman et al., 2010) is a policy enforcement framework that allows a user imposes permission constraints on the use of resources. Peng et al. (2012) and Sarma et al. (2012) use permissions requested by an application to identify the risk of the applications. WHYPER (Pandita et al., 2013) uses natural language processing (NLP) technology to analyse the application description and identify the need of permissions to the application. However, the semantic gap between permissions and resources makes it only identify a few permissions. A lightweight possession proof scheme has been proposed which is based on chameleon hash function (Ren and Liu, 2014). AppAware relies on the set of permissions exposed by each application to detect the malicious applications. Pscout (Au et al., 2012) is a static tool that extracts the permission specification from the source code of Android. VetDroid (Zhang et al., 2013) proposed a dynamic analysis platform that analyses sensitive behaviours based on perspective of permission use on Android, but it cannot cover all of user permissions in the application. Sang et al. (2014) propose a security evaluation of smartphone operating system based on international security assessment criteria.

Many researchers proposed using machine learning methods to detect malwares on Android. Although their methods achieved a better detection rate, the false positive ratio will increase while increasing the number of permissions. We find that malicious applications request more dangerous permissions and receive more system event messages. We extend existing permission-based risk analysis methods, add broadcast receivers as feature vectors, and propose IPBD method to identify the malicious applications. Experimental results show that our IPBD method has a high accuracy rate, and the false positive ratio will not increase when the number of permissions grows.

7 Conclusions

We analysed existing permission-based risk analysis methods and the characteristics of the malware app behaviours on Android. Although there are many permission-based approaches to identify malwares, none of them gives attention to how the malwares activate their malicious components. We introduced the Android system event messages as feature vectors, and proposed IBD and IPBD methods to identify the risks of applications for mobile users. The methods not only analyse the dangerous permissions of the applications, but also focus on which system broadcast will be received by the applications. This makes our methods more difficult to be evaded compared with other permission-based methods. We compare our methods with other related work on real world datasets to test their ability of revealing the malicious applications. Experimental results show that the IPBD method is quite effective, which has high accuracy in identifying malicious applications and a low rate of false positives in practice. This means that the proposed method can distinguish benign applications which request many dangerous permissions from malwares. As a consequence, the IPBD method can be regarded as a fast filter to identify malicious applications for applications markets. The proposed approaches are based on the key insight that most of the malwares samples request plenty of dangerous permissions and include malicious payloads by adding broadcast receiver components and service components. They are not appropriate for the malwares that inject their malicious behaviour in the activities or multiply applications collusive attack, because those malwares need not use broadcast to activate components or request many permissions. Next, we take broadcast received and service as entry points to analyse the malicious behaviours.

Acknowledgements

This research is partially supported by National Natural Science Foundation of China under grants 61173170, 61300222, 61433006, and U1401258, Innovation Fund of Huazhong University of Science and Technology under grants 2013QN120, 2012TS052 and 2012TS053.

References

Ahamad, S.S., Sastry, V. and Udgata, S.K. (2014) 'Secure mobile payment framework based on UICC with formal verification', *International Journal of Computational Science and Engineering*, Vol. 9, No. 4, pp.355–370.

Au, K.W.Y., Zhou, Y.F., Huang, Z. and Lie, D. (2012) 'PScout: analyzing the Android permission specification', *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ACM.

Barrera, D., Kayacik, H.G., van Oorschot, P.C. and Somayaji, A. (2010) 'A methodology for empirical analysis of permission-based security models and its application to android', *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ACM.

Bartel, A., Klein, J., Le Traon, Y. and Monperrus, M. (2012) 'Automatically securing permission-based software by reducing the attack surface: an application to Android', *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, ACM.

Blasing, T., Batyuk, L., Schmidt, A-D., Camtepe, S.A. and Albayrak, S. (2010) 'An android application sandbox system for suspicious software detection', *2010 5th International Conference on Malicious and Unwanted Software (MALWARE)*, IEEE.

Burguera, I., Zurutuza, U. and Nadjm-Tehrani, S. (2011) 'Crowdroid: behavior-based malware detection system for android', *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, ACM.

Chakradeo, S., Reaves, B., Traynor, P. and Enck, W. (2013) 'Mast: triage for market-scale mobile malware analysis', *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ACM.

Chang, C-C. and Lin, C-J. (2011) 'LIBSVM: a library for support vector machines', *ACM Transactions on Intelligent Systems and Technology (TIST)*, Vol. 2, No. 3, p.27.

Chen, H-C., Huang, Y-F., Lee, S-H., Chen, C-T. and Hung, H-L. (2012) 'A mobile location-based with time-constraint RBAC associated database management model', *International Journal of Computer Systems Science & Engineering*, Vol. 27, No. 6, pp.431–440.

Chin, E., Felt, A.P., Greenwood, K. and Wagner, D. (2011) 'Analyzing inter-application communication in Android', *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ACM.

Davi, L., Dmitrienko, A., Sadeghi, A-R. and Winandy, M. (2011) 'Privilege escalation attacks on android', *Information Security*, Springer, pp.346–360.

Enck, W., Gilbert, P., Chun, B-G., Cox, L.P., Jung, J., McDaniel, P. and Sheth, A. (2010) 'TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones', *OSDI*.

Enck, W., Ongtang, M. and McDaniel, P. (2009) 'On lightweight mobile phone application certification', *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ACM.

Felt, A.P., Chin, E., Hanna, S., Song, D. and Wagner, D. (2011a) 'Android permissions demystified', *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ACM.

Felt, A.P., Finifter, M., Chin, E., Hanna, S. and Wagner, D. (2011b) 'A survey of mobile malware in the wild', *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, ACM.

Felt, A.P., Greenwood, K. and Wagner, D. (2011c) 'The effectiveness of application permissions', *Proceedings of the 2nd USENIX Conference on Web Application Development*, USENIX Association.

Grace, M., Zhou, Y., Zhang, Q., Zou, S. and Jiang, X. (2012) 'Riskranker: scalable and accurate zero-day android malware detection', *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ACM.

Han, J. and Kamber, M. (2006) *Data Mining: Concepts and Techniques*, 2nd ed., Elsevier Science & Technology, Burlington, MA, USA.

Hartigan, J.A. and Wong, M.A. (1979) 'Algorithm AS 136: a k-means clustering algorithm', *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 28, No. 1, pp.100–108.

Huang, C-Y., Tsai, Y-T. and Hsu, C-H. (2013) 'Performance evaluation on permission-based detection for Android malware', in J-S. Pan, C-N. Yang and C-C. Lin (Eds.): *Advances in Intelligent Systems and Applications – Volume 2*, Vol. 21, pp.111–120, Springer Berlin Heidelberg.

Jeon, J., Micinski, K.K., Vaughan, J.A., Fogel, A., Reddy, N., Foster, J.S. and Millstein, T. (2012) 'Dr. Android and Mr. Hide: fine-grained permissions in android applications', *Proceedings of the second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, ACM.

Jiang, X. (2012) *An Evaluation of the Application ('App') Verification Service in Android 4.2*, North Carolina State University.

Lowe, D.G. (1999) 'Object recognition from local scale-invariant features', *The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999*, IEEE.

Magat, W.A., Viscusi, W.K. and Huber, J. (1988) 'Consumer processing of hazard warning information', *Journal of Risk and Uncertainty*, Vol. 1, No. 2, pp.201–232.

Nauman, M., Khan, S. and Zhang, X. (2010) 'Apex: extending android permission model and enforcement with user-defined runtime constraints', *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ACM.

Pandita, R., Xiao, X., Yang, W., Enck, W. and Xie, T. (2013) 'WHYPER: towards automating risk assessment of mobile applications', *Proceedings of the 22nd USENIX Security Symposium*, Washington DC, USA.

Peng, H., Gates, C., Sarma, B., Li, N., Qi, Y., Potharaju, R., Nita-Rotaru, C. and Molloy, I. (2012) 'Using probabilistic generative models for ranking risks of Android apps', *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ACM.

Rastogi, V., Chen, Y. and Jiang, X. (2013) 'DroidChameleon: evaluating Android anti-malware against transformation attacks', *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ACM.

Ren, W. and Liu, Y. (2014) 'A lightweight possession proof scheme for outsourced files in mobile cloud computing based on chameleon hash function', *International Journal of Computational Science and Engineering*, Vol. 9, No. 4, pp.339–346.

Sang, J., Hong, D., Zhang, B., Xiang, H. and Fu, L. (2014) 'Protection profile for the smartphone operating system', *International Journal of Embedded Systems*, Vol. 6, No. 1, pp.28–37.

Sarma, B.P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C. and Molloy, I. (2012) 'Android permissions: a perspective combining risks and benefits', *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, ACM.

Schmidt, A-D., Bye, R., Schmidt, H-G., Clausen, J., Kiraz, O., Yuksel, K.A., Camtepe, S.A. and Albayrak, S. (2009) 'Static analysis of executables for collaborative malware detection on android', *IEEE International Conference on Communications, 2009, ICC'09*, IEEE.

Schmidt, A-D., Schmidt, H-G., Clausen, J., Yuksel, K.A., Kiraz, O., Camtepe, A. and Albayrak, S. (2008) 'Enhancing security of linux-based android devices', in *Proceedings of 15th International Linux Kongress*, Lehmann.

Security, L.M. (2011.3) *Lookout Mobile Security Technical Tear Down – droiddream* [online] https://blog.lookout.com/wp-content/uploads/2011/03/COMPLETE-DroidDream-Technical-Tear-DownLookout-Mobile-Security.pdf (accessed 20 February 2014).

Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C. and Weiss, Y. (2012) 'Andromaly: a behavioral malware detection framework for android devices', *Journal of Intelligent Information Systems*, Vol. 38, No. 1, pp.161–190.

Shastry, A., Kantarcioglu, M., Zhou, Y. and Thuraisingham, B. (2012) 'Randomizing smartphone malware profiles against statistical mining techniques', *Data and Applications Security and Privacy XXVI*, Springer, pp.239–254.

Shimada, H., Courbot, A., Kinebuchi, Y. and Nakajima, T. (2011) 'A software infrastructure for dependable embedded systems', *International Journal of Computer Systems Science & Engineering*, Vol. 26, No. 6, pp.491–503.

Traynor, P., Lin, M., Ongtang, M., Rao, V., Jaeger, T., McDaniel, P. and La Porta, T. (2009) 'On cellular botnets: measuring the impact of malicious devices on a cellular network core', *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ACM.

Wei, X., Gomez, L., Neamtiu, I. and Faloutsos, M. (2012) 'Permission evolution in the Android ecosystem', *Proceedings of the 28th Annual Computer Security Applications Conference*, ACM.

Yan, L.K. and Yin, H. (2012) 'Droidscope: seamlessly reconstructing the OS and Dalvik semantic views for dynamic android malware analysis', *Proceedings of the 21st USENIX Security Symposium*.

Zhang, Y., Yang, M., Xu, B., Yang, Z., Gu, G., Ning, P., Wang, X.S. and Zang, B. (2013) 'Vetting undesirable behaviors in android apps with permission use analysis', *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ACM.

Zhongyang, Y., Xin, Z., Mao, B. and Xie, L. (2013) 'DroidAlarm: an all-sided static analysis tool for Android privilege-escalation malware', *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ACM.

Zhou, W., Zhou, Y., Jiang, X. and Ning, P. (2012) 'Detecting repackaged smartphone applications in third-party android marketplaces', *Proceedings of the second ACM conference on Data and Application Security and Privacy*, ACM.

Zhou, Y. and Jiang, X. (2012) 'Dissecting android malware: characterization and evolution', *2012 IEEE Symposium on Security and Privacy (SP)*, IEEE.